Copy No._____

DEFENCE R&D DÉFENSE

A Practical Guide to Level One Data Fusion Algorithms

D.J. Peters

Defence R&D Canada

Technical Memorandum

DREA TM 2001-201

December 2001

National Defence Défense nationale

Canada

A Practical Guide to Level One Data Fusion Algorithms

D. J. Peters

Defence Research Establishment Atlantic

Technical Memorandum

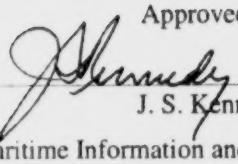
DREA TM 2001-201

December 2001

Author

D. J. Peters

Approved by



J. S. Kennedy

Head, Maritime Information and Combat Systems Section

Approved for release by

Kirk Foster

DRP Chair

© Her Majesty the Queen as represented by the Minister of National Defence, 2001

© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2001

104-06915/

Abstract

Level one data fusion is the process of combining data in order to track and classify individual entities. This document introduces the basic concepts and presents a core selection of standard algorithms, such as the Kalman Filter, the Interacting Multiple Model (IMM) filter, the Probabilistic Data Association Filter (PDAF) and its Joint variant, the Munkres algorithm for Nearest Neighbour (NN) association, and Multiple-Hypothesis Tracking (MHT), among others. It is intended to serve as a convenient one-stop repository of algorithms.

Résumé

La fusion de données de niveau un est le processus qui consiste à combiner des données afin de pister et de classer des entités particulières. Ce document présente les concepts de base et une sélection d'algorithmes standard essentiels de fusion de données, comme le filtre de Kalman, le filtre du modèle multiple avec interactions (Interacting Multiple Model – IMM), le filtre avec association probabiliste des données (Probabilistic Data Association Filter – PDAF) et sa variante conjointe (JPDAF), l'algorithme de Munkres pour l'association avec le plus proche voisin (Nearest Neighbour – NN) et le pistage de données fondé sur plusieurs hypothèses (Multiple-Hypothesis Tracking – MHT) entre autres. Le document vise à présenter en un seul endroit un ensemble d'algorithmes.

This page intentionally left blank.

Executive summary

Operators of modern military platforms and systems are being increasingly inundated with target information, particularly in cluttered littoral environments. With multiple sensors and platforms often covering the same region of space and time, the profusion of contact and track data for a given target can add considerable confusion to the picture the command team is trying to build of the battle space. Automated data fusion methods can be used to combine these multiple contacts and tracks into a smaller number of tracks, thus simplifying the process of the formation of the tactical picture. Ideally, the resulting tracks represent an optimized treatment of all available sensor and link data. Automated data fusion, as a way of managing a potentially large amount of sensor and link data, is therefore an important enabler for military decision making.

There are several levels of abstraction in data fusion, but we focus here on "level one", in which data are combined in order to track and classify individual moving targets, without reference to any abstract relationships among the targets or to the possible conscious intentions of a target. The data to be fused may be post-detection sensor data (contacts) or prior output from level one data fusion (tracks).

This document is intended to serve as a convenient one-stop repository of what the author has judged to be a core selection of standard data fusion algorithms. These include, among others, the Kalman Filter, the Interacting Multiple Model (IMM) filter, the Probabilistic Data Association Filter (PDAF) and its Joint variant, the Munkres algorithm for Nearest Neighbour (NN) association, and Multiple-Hypothesis Tracking (MHT), along with some track-level fusion methods such as Covariance Intersection (CI). It is believed that the more complicated algorithms, such as the MHT, will be found to have greater clarity as presented here than they have in the references from which they were gathered.

The algorithms are presented with enough context that a beginner will be helped in understanding when and how to apply them. Nevertheless, the presentation of the fundamental concepts that are necessary in order to appreciate the algorithms has been kept brief, and readers are directed elsewhere for a more thorough presentation of the purposes of data fusion and its areas of applicability.

Peters, D. J. 2001. A Practical Guide to Level One Data Fusion Algorithms.
DREA TM 2001-201 Defence Research Establishment Atlantic.

Sommaire

Les opérateurs des plates-formes et systèmes militaires contemporains sont submergés par un volume croissant d'informations sur les cibles, spécialement dans les environnements littoraux encombrés. Souvent, la même région est observée par plusieurs capteurs et plates-formes simultanément, ce qui engendre une profusion de données sur les contacts et les trajectoires, et produit une grande confusion dans l'image de l'espace de combat que l'équipe de commandement essaie de se créer. On peut utiliser des méthodes automatiques de fusion de données pour réduire ces nombreux contacts et trajectoires à un nombre restreint de trajectoires, ce qui simplifie la création de l'image tactique. Théoriquement, les trajectoires résultantes proviendraient du traitement optimisé de toutes les données de capteurs et de toutes les données de liaison qui sont disponibles. Parce qu'elle permet de gérer un nombre potentiellement grand de données de capteurs et de liaison, la fusion automatique de données facilite grandement la prise de décisions militaires.

Il existe plusieurs niveaux d'abstraction pour la fusion de données. Nous ne nous penchons ici que sur le « premier niveau », dans lequel les données sont combinées afin de permettre le suivi et le classement de chacune des cibles mobiles, sans tenter d'établir des relations abstraites entre les cibles ou de deviner leurs intentions conscientes probables. Les données à fusionner peuvent être des données obtenues après détection par un capteur (contacts) ou le résultat d'une fusion antérieure de niveau un de données (trajectoires).

Ce document vise à présenter en un seul endroit un ensemble de ce que l'auteur a jugé être une sélection d'algorithmes standard essentiels de fusion de données. Ces algorithmes comprennent, entre autres, le filtre de Kalman, le filtre du modèle multiple avec interactions (Interacting Multiple Model – IMM), le filtre avec association probabiliste des données (Probabilistic Data Association Filter – PDAF) et sa variante conjointe (JPDAF), l'algorithme de Munkres pour l'association avec le plus proche voisin (Nearest Neighbour – NN) et le pistage de données fondé sur plusieurs hypothèses (Multiple-Hypothesis Tracking – MHT), ainsi que certaines méthodes de fusion au niveau de la trajectoire, comme l'intersection de covariances (Covariance Intersection – CI). On croit que l'on jugera que les algorithmes les plus complexes, comme le MHT, sont présentés de façon plus claire ici que dans les références d'où ils ont été tirés.

Les algorithmes sont présentés avec suffisamment de contexte pour aider un débutant à comprendre quand et comment les appliquer. Néanmoins, la présentation des concepts fondamentaux qui sont nécessaires pour apprécier l'algorithme a été résumée et les lecteurs sont dirigés ailleurs pour obtenir une présentation plus complète des buts de la fusion de données et des domaines dans lesquels on peut l'appliquer.

Peters, D. J. 2001. A Practical Guide to Level One Data Fusion Algorithms.
DREA TM 2001-201 Defence Research Establishment Atlantic.

Table of contents

Abstract i

Résumé i

Executive summary iii

Sommaire iv

Table of contents v

List of figures vii

1. Introduction 1
    1.1 Data Fusion Roughly Described 1
    1.2 Contact Data Versus Track Data 2
    1.3 Outline of the Following Chapters 4

2. Basic Single-Target Tracking 5
    2.1 A Simple Example of Recursive Estimation 5
    2.2 The Kalman Filter 6
    2.3 The Extended Kalman Filter 10
    2.4 The Rectangular-Polar Problem 11
    2.5 The Interacting Multiple Model Filter 13
    2.6 Summary 15

3. False Alarms 16
    3.1 Gates and Validation 16
    3.2 Nearest-Neighbour and Strongest-Neighbour Methods 18
    3.3 The Probabilistic Data Association Filter 18
    3.4 Summary 21

4. Track Initiation and Deletion 23
    4.1 "2/2 & m/n" 23
    4.2 Track Initiation in the IMMPDAF 25

4.3 Single-Measurement Track Initiation 26
4.4 Summary 27

5. Multiple Targets 28
5.1 Nearest-Neighbour Methods 28
5.2 The Joint Probabilistic Data Association Filter 30
5.3 Multiple Hypothesis Tracking 32
5.4 Summary 36

6. Identity Data Fusion 37
6.1 Basic Probability Assignments 37
6.2 Dempster's Rule 39
6.3 Fusing Identity Data With Kinematic Data 39

7. Multiple Sensors 41
7.1 Data Fusion Architectures 41
7.2 Out-of-Sequence Measurements 43
7.3 Track-Level Fusion 44

8. Other Issues in Level One Data Fusion 51

9. Conclusion 53

10. References 54

Annexes 55

Distribution List 58

List of figures

Figure 1. A centralised architecture 42

Figure 2. A sensor-based hierarchical architecture with feedback 42

Figure 3. A hybrid hierarchical architecture without feedback 42

Figure 4. A sensor-based distributed architecture 42

Figure 5. The formation of ghosts in triangulation 51

1. Introduction

This document presents a core selection of data fusion algorithms. The algorithms are presented with enough context so that a beginner will be helped in understanding when and how to apply them, and in most cases with enough detail so that a programmer could create code for them (provided attention is paid also to numerical stability, a topic of which little is said here). The primary objective of this document is to serve as a convenient one-stop repository of basic algorithms, and to add clarity to their presentation. Such clarification is potentially useful because the references from which the algorithms are taken (usually [1] or [2]) present some of these algorithms (especially the more complicated ones) in a fragmentary manner, or with inconsistent or ambiguous notation. On the other hand, the present document, unlike some of the references, does not provide any mathematical justification.

For a general discussion of the purposes of data fusion, its areas of applicability, and the kinds of data that can be fused, see [3]. In this document the emphasis is on problems related to the tracking and identification of moving targets, with a military setting usually assumed. The limited focus of this document will be further explained in section 1.1, which also serves to present all that will be said here about the broader view of data fusion. Section 1.2 introduces some concepts that are necessary for all that follows in later chapters, while section 1.3 briefly outlines the plan of the rest of the document.

1.1 Data Fusion Roughly Described

Data[1] *fusion* can be defined as "the process of combining data to refine state estimates and predictions" [4]. In a military context, the state estimates and predictions in question consist of the identity, position, status, and behaviour of objects that may have an impact on the success of the mission.

The data to be fused can come from any imaginable source (or set of sources), but we will concentrate on data from sensors (such as radar) that have some ability to localize the source of the energy being sensed. The need for data fusion arises from the increasing number and sophistication of the sensing devices available to modern military forces, and the resulting high rate of incoming data. Hence data fusion can be thought of as the effective management of data in order to maximize their usefulness.

The end product of data fusion can be presented or envisioned as a "picture". Indeed, when data fusion is carried out on a tactical scale, the end product is referred to as a *tactical picture*. We are concerned here with tactical picture compilation.

The process of data fusion can occur at several different levels of abstraction. It is customary to divide and label the levels according to the following scheme [3] [4]:

- Level one data fusion, sometimes called *object assessment*, is the tracking and classification of individual objects.

[1] No distinction between *data* and *information* will be made in this document.

- Level two data fusion, also referred to as *situation assessment* (SA), seeks to provide further understanding of objects' behaviour by examining the behaviour of other objects, in the context of the physical surroundings. SA is thus concerned with the relationships among objects, and between objects and the environment.

- Level three data fusion, also referred to as *impact assessment*, estimates and predicts the effects of planned or anticipated actions by the participants. In a military setting, where the term *threat assessment* (TA) is also used for this level of data fusion, the focus is on judging the level of danger, inferring enemy intent, and identifying opportunities for action.

- Level four data fusion is the refinement of the data fusion process itself. For example, one might adjust the behaviour of one's sensors, or adjust the parameters of one's data fusion algorithms, in response to the output of the first three levels. This *process refinement* is part of *resource management* (RM), which refers both to level four data fusion and to other kinds of process-related decisions that are not part of data fusion (such as the firing of weapons).

This document is concerned with level one data fusion. It is assumed that we need not concern ourselves with sensor design or signal processing. For our purposes, a sensor either believes it has detected an object or it does not. If it does, it gives us data relating to the nature or the behaviour of that alleged object[2].

1.2 Contact Data Versus Track Data

A *contact* is the package of data that is derived from a single detection, without reference to other detections. It can be thought of, roughly, as a single glimpse of a possible target[3]. A contact should include a time tag (telling the time when the detection took place) and whatever data are available to the sensor about the alleged target at that time, along with the degree of uncertainty associated with each of these data. The term *measurement* will be used to refer to the mathematical value of whatever part of a contact is used in a given fusion algorithm. In most of this document, a measurement will be a position of a target, expressed as a vector, assumed to be taken directly from contact data.

In general it is not obvious which contact corresponds to which target, or even whether a given contact corresponds to anything real at all. If a contact does correspond to something real, then it should be possible to sense that target again and again. Therefore, a contact that is not corroborated by another contact is discarded, while multiple contacts that are deemed to belong (probably) to the same target are fused together into a *track*. A track is a data package that is considered to refer to a single target, is derived from contacts (usually two or more), and can be occasionally updated with reference to further contacts. It will contain data of the

---

[2] In practice, the response of a sensor (or a combination of responses from one or multiple sensors) will be periodically compared with some pre-set threshold. If the threshold is exceeded, we say that a detection has occurred. The timing of the detection attempts depends on the design and construction of the sensor and in some cases on the choice of signal processing strategy. Arguably, the term *data fusion* is defined broadly enough to include pre-detection processing. The term *sub-object data assessment* has been proposed to refer to this "level zero" data fusion.

[3] All objects of interest are referred to as *targets*. This does not imply any intention to shoot at them.

same general type as those contained in a contact, as well as data that could not have appeared in any of the individual contacts but is inferred from the contacts collectively. For example, a track may contain an estimate of a target's velocity, derived from contacts that contain only position data. In addition, a track will contain some self-referential data such as the time when it was initiated, the time when it was last updated, and some kind of overall score reflecting the system's confidence that the track represents a real target.

A track has a life cycle of *initiation*, *maintenance*, and *deletion*. Track initiation occurs when contacts appear that are believed to belong to a target, and it is believed that the target in question is not already represented by a track. A track is maintained while contacts appear that are believed to belong to the target represented by that track, and the track's data are updated with reference to those contact data. A track is deleted when, for a significant period of time, no such new contacts appear. This may occur for a variety of reasons. Perhaps the target has become invisible to the reporting sensor, as a result of being destroyed or moving beyond the sensor's range, or as a result of changing environmental conditions. Perhaps (in the case of a short-lived track) it was never a real target at all. Another possibility is that the target changed its behaviour in such a way that further contacts from that target were mistakenly considered to belong to a different target.

Contact data may arrive at a high rate. It is not practical, in general, to store all contacts and repeatedly refer to all past contacts for the sake of a statistically optimal treatment of all available data. For this reason a contact is not usually stored beyond the point when it is used to initiate or update a track. The updating of a track, as a result of assigning a new contact to it, is done with reference to the track's old data (with their degrees of uncertainty) and the contact's data (with their degrees of uncertainty). The track's new data are derived as a compromise between its old data and the contact's data. With respect to the continuous-valued characteristics of a target, such as its position and velocity, this compromise takes the form of a weighted average between a measured value and a value that is predicted according to past data. The question of how much weight to assign to the measured and predicted values is central to level one data fusion. For this purpose, the Kalman Filter (section 2.2) forms the basis of our set of tools.

Before those decisions about averaging can be made, one must address the matter of *association*, which is the question of which contacts or tracks are to be identified with each other, in the sense that they are believed to refer to the same target. Contacts must be assigned to other contacts in order to initiate a track, and contacts must be assigned to tracks in order for those tracks to be maintained. Also, in practice, the formation of tracks is often done with reference to a single sensor, so that it is also necessary to be able to assign tracks to other tracks.

Within level one data fusion, we may distinguish between *contact-level* and *track-level* fusion. Contact-level fusion uses contact data (or their absence) to initiate, maintain, and delete tracks. In track-level fusion, tracks from multiple sources are used to maintain each other or to form new tracks. The tracks that are the inputs for track-level fusion are themselves the outputs of some fusion process that may be either contact-level or track-level. Track-level fusion is therefore a recursive step in an overall process of fusing contact data into tracks.

1.3 Outline of the Following Chapters

Chapter 2 will address the very limited situation of track maintenance for a single target, dealing with kinematic variables only, measured by a single sensor, with no decisions to be made regarding association. Chapter 3 will introduce the question of association as it applies to distinguishing between real targets and false alarms. Chapter 4 will introduce the initiation and deletion of tracks, and Chapter 5 expands the association and track maintenance questions to the case of multiple targets. Each of Chapters 2 through 5 includes a short summary at the end, identifying the capabilities and limitations of each of the methods presented. Chapter 6 discusses non-kinematic data. Chapter 7 considers multiple sensors, and is the last chapter for which any amount of detail is provided. Chapter 8 provides a partial list of issues relevant to level one data fusion that are not dealt with in this document.

2. Basic Single-Target Tracking

Throughout this chapter we will be concerned only with the updating (maintenance) of a single track that is assumed to be already initiated. As new contacts arrive, they are all used to update the state of the track, then discarded. This calls for a recursive technique. The most popular technique is the *Kalman Filter*.

2.1 A Simple Example of Recursive Estimation

Before describing the Kalman Filter, let us consider the following toy problem: Suppose we are making repeated measurements x_i of a single fixed quantity x, that the error in each measurement is normally distributed with mean zero and variance σ^2 (same for each measurement), and that the errors in the measurements are mutually independent. Not surprisingly, the best estimate of x is simply the mean of the measurements:

$$\hat{x}_N = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad 2.1\text{-}1$$

(where N is the total number of measurements so far) while the variance associated with the estimate is σ^2 / N. (It is common in tracking theory for estimates to be denoted with a caret, as $\hat{x}$, for example; here we are using $\hat{x}_N$ to mean "our estimate of x based upon the first N measurements".) In order to recast this as a recursive problem, let us suppose that, having already accumulated N measurements and made an estimate accordingly, we then get an $(N+1)$th measurement. There is no need to recalculate the mean from scratch, since the new mean can be calculated from the old mean and the new measurement by:

$$\hat{x}_{N+1} = \frac{N}{N+1}\hat{x}_N + \frac{1}{N+1}x_{N+1} = \hat{x}_N + \frac{1}{N+1}\left(x_{N+1} - \hat{x}_N\right) \qquad 2.1\text{-}2$$

Clearly this procedure can be continued *ad infinitum* as more measurements are made. Keeping in mind our overall objective of discussing data fusion, we can think of each measurement as a contact, and each successive estimate as the updated track. Note that each contact can be discarded after it has been duly applied to the task of updating the track.

The expression on the right is typical of an updated estimate. The difference between the latest measurement and the previous estimate is multiplied by a factor called the *gain*, here equal to $1/(N+1)$, and this is added to the old estimate to make the new one. An important issue in tracking is deciding what is the appropriate gain when updating an estimate, as the gain represents the degree to which the measurement is emphasized in comparison with the prior estimate.

In order to make our toy problem fully recursive, we should eliminate the need to keep track of the number of measurements made so far. (To emphasize the recursive nature of the technique, the estimates will be denoted $\hat{x}_{\text{this}}$ and $\hat{x}_{\text{next}}$; no numeric subscript is necessary.) We can accomplish this by keeping, along with the current estimate, the variance

(uncertainty) associated with it, which will be denoted P_{this} or P_{next}, as appropriate. The new measurement, x_{next}, takes us from "this" state to the "next" state thus:

$$\hat{x}_{\text{next}} = \hat{x}_{\text{this}} + \frac{P_{\text{this}}}{P_{\text{this}} + \sigma^2}\left(x_{\text{next}} - \hat{x}_{\text{this}}\right)$$
$$P_{\text{next}} = \frac{P_{\text{this}}\sigma^2}{P_{\text{this}} + \sigma^2}. \qquad 2.1\text{-}3$$

If we take the initial estimate to be equal to the initial measurement, and the initial estimate variance to be equal to the measurement variance σ^2, then it is not hard to show that these formulae lead to the estimate always being equal to the mean of the measurements taken thus far and the estimate variance always being equal to σ^2 / N, as it ought to be. Better yet, if the measurement variances are themselves variable, then these formulae will lead to the appropriately weighted average value of x (i.e., the statistically "expected" value) along with the correct variance of that estimate, if we simply replace σ by σ_{next}, i.e., the measurement variance associated with the new measurement.

2.2 The Kalman Filter

Now we are ready to discuss the Kalman Filter (KF). Despite the name "Filter", the KF is really an algorithm for recursive estimation. The final formulae of our toy problem above constitute a special case of the KF. In the toy problem, the quantity we were estimating was a constant. We would like to be able to make an estimate of a quantity that changes in time (such as the position of an aircraft) based upon a series of measurements. A KF can handle a wide variety of types of time-dependence. In order to apply the technique, we must express the motion of the target as a combination of predictable and unpredictable components. More precisely, we model the target behaviour as

$$\mathbf{x}(k+1) = \mathbf{F}(k)\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) + \mathbf{v}(k) \qquad 2.2\text{-}1$$

where the various pieces of this *process model* are described in the following paragraphs.

The state of the target is represented by the n-dimensional (column) vector $\mathbf{x}(k)$. As an example, for the sake of tracking an aircraft, we might use a six-dimensional vector with the cartesian coordinates of its position and velocity:

$$\mathbf{x}(k) = \begin{bmatrix} x(k) & y(k) & z(k) & \dot{x}(k) & \dot{y}(k) & \dot{z}(k) \end{bmatrix}^{\mathrm{T}} \qquad 2.2\text{-}2$$

(the superscript T indicates the matrix transpose operation). The index k in the model is a discrete time index. The amount of time elapsed between one sample and the next does not have to be constant (although it is customary in many applications to keep it constant); all that matters here is that we are counting forwards, so that $k+1$ represents the next sampling time after k.

The $n \times n$ matrix $\mathbf{F}(k)$ is the *transition matrix*, which represents how the state of the target would change "ideally"; i.e., left on its own and (more importantly) in the absence of the unpredictable factors. Extending the six-dimensional example of the state vector above, the transition matrix would be:

$$\mathbf{F}(k) = \begin{bmatrix} \mathbf{I}_3 & T\mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}. \qquad 2.2\text{-}3$$

The transition matrix of this example would be sufficient on its own to describe the motion of our target if it were not accelerating at all. Here we are assuming only that the target is not *steadily* accelerating. The effects of "unsteady" acceleration will be swept into the $\mathbf{v}(k)$ term. Here T is the amount of time elapsed between k and $k+1$, $\mathbf{I}_3$ is the 3×3 identity matrix, and $\mathbf{0}_3$ is the 3×3 zero matrix.

The vector $\mathbf{u}(k)$ represents whatever degree of control we have over the target state. For example, if we were using the position of our sensor as the origin of the coordinate system in which we consider the target's position, then the sensor's motion will have an impact on the target's position. In this case the sensor's position and velocity could be put together into $\mathbf{u}(k)$. The matrix $\mathbf{G}(k)$ does whatever is necessary to correctly apply the effects of $\mathbf{u}(k)$. The $\mathbf{G}(k)\mathbf{u}(k)$ term is included in equation 2.2-1 for the sake of completeness, but it will usually be dropped from now on.

The last term, $\mathbf{v}(k)$, is the *process noise* term. Its components are random variables with zero mean. In order for the KF to work properly, the process noise should be Gaussian (that is, normally distributed) and "white" (which is to say that its value at one time is unrelated to its value at another time). The covariance matrix

$$\mathbf{Q}(k) \equiv \left\langle [\mathbf{v}(k) - \bar{\mathbf{v}}(k)][\mathbf{v}^{\mathrm{T}}(k) - \bar{\mathbf{v}}^{\mathrm{T}}(k)] \right\rangle = \left\langle \mathbf{v}(k)\mathbf{v}^{\mathrm{T}}(k) \right\rangle \qquad 2.2\text{-}4$$

is considered to be known. (Here $\bar{\mathbf{v}}(k)$, the mean of $\mathbf{v}(k)$, is zero. The angular brackets denote the mean (statistically expected value) of the expression contained therein.) The diagonal elements of $\mathbf{Q}(k)$ are the variances of the components of $\mathbf{v}(k)$, while the off-diagonal elements provide a measure of the correlations among the components of $\mathbf{v}(k)$. Continuing our aircraft-tracking example, if we consider the process noise to result in random changes to the target's velocity, and the changes to the three velocity components to be mutually uncorrelated, then a realistic form of the covariance matrix might be:

$$\mathbf{Q}(k) = \begin{bmatrix} \frac{T^3}{3}\mathbf{I}_3 & \frac{T^2}{2}\mathbf{I}_3 \\ \frac{T^2}{2}\mathbf{I}_3 & T\mathbf{I}_3 \end{bmatrix} \frac{\sigma_a^2}{f_{ac}} \qquad 2.2\text{-}5$$

where σ_a^2 is the assumed variance in the acceleration (some people use $\sigma_a^2 = a_{\max}^2/9$ where $a_{\max}$ is the maximum acceleration of which we believe the target to be capable), and f_{ac} can be thought of as the frequency with which decisions to change the acceleration would be made (e.g., by the pilot), perhaps in the order of 1 Hz. See [5] for an explanation[4] of 2.2-5 as well as other possible forms of a process noise covariance matrix.

---

[4] The derivation assumes that the acceleration appears as white noise, and therefore that the expected value of a product of one component of the acceleration at one time and the same component of the acceleration at another time is a Dirac delta function of the difference of the times: $\langle a(t)a(t')\rangle = q\delta(t-t')$ for some constant q. To get equation 2.2-5 we set $q = \sigma_a^2 / f_{ac}$. Note

So now we have a process model. We also need a *measurement model*:

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{w}(k). \qquad 2.2\text{-}6$$

$\mathbf{z}(k)$ is an m -dimensional vector representing our measurement of the target at the time labeled by k. The $m \times n$ *measurement matrix* $\mathbf{H}(k)$ relates the target state to an ideal (error-free) measurement. To continue our aircraft tracking example, if we are measuring the position (only), in cartesian coordinates (admittedly unlikely), then

$$\mathbf{H}(k) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix}. \qquad 2.2\text{-}7$$

The m -dimensional vector $\mathbf{w}(k)$ is the *measurement noise*, representing the inevitable errors in our measurement. Its covariance matrix $\mathbf{R}(k)$, defined in terms of $\mathbf{w}(k)$ the same way that $\mathbf{Q}(k)$ is defined in terms of $\mathbf{v}(k)$, is considered to be known. In our ongoing example, if we have the same uncertainty in each cartesian component, and these uncertainties are not mutually correlated, then we would have $\mathbf{R}(k) = \sigma_m^2 \mathbf{I}_3$, where σ_m^2 is the variance associated with each component of the measurement.

The assumptions behind the KF are now in place. The only thing left to do before presenting the technique in all its glory is to introduce some of the peculiar notation that is commonly used in estimation theory. $\hat{\mathbf{x}}(k \mid k')$ denotes the estimate of $\mathbf{x}(k)$ taking into account all the measurements taken up to and including $\mathbf{z}(k')$. If $k' < k$ then this estimate is a prediction based on prior measurements. Associated with each estimate is a covariance matrix $\mathbf{P}(k \mid k')$ representing the uncertainty in the estimate. If the chosen process model is "good" (this will be discussed below), then

$$\mathbf{P}(k \mid k') = \left\langle [\mathbf{x}(k) - \hat{\mathbf{x}}(k \mid k')][\mathbf{x}^{\mathrm{T}}(k) - \hat{\mathbf{x}}^{\mathrm{T}}(k \mid k')] \right\rangle. \qquad 2.2\text{-}8$$

The KF provides a way to get from $\hat{\mathbf{x}}(k-1 \mid k-1)$ and $\mathbf{P}(k-1 \mid k-1)$ to $\hat{\mathbf{x}}(k \mid k)$ and $\mathbf{P}(k \mid k)$ using the new measurement $\mathbf{z}(k)$. The KF algorithm can be stated this way (this is based on [1]):

$$\begin{aligned}
&\mathbf{P}(k \mid k-1) = \mathbf{F}(k-1)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^{\mathrm{T}}(k-1) + \mathbf{Q}(k-1) \\
&\mathbf{S}(k) = \mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^{\mathrm{T}}(k) + \mathbf{R}(k) \\
&\mathbf{W}(k) = \mathbf{P}(k \mid k-1)\mathbf{H}^{\mathrm{T}}(k)\mathbf{S}^{-1}(k) \\
&\mathbf{P}(k \mid k) = \mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^{\mathrm{T}}(k) \qquad 2.2\text{-}9 \\
&\hat{\mathbf{x}}(k \mid k-1) = \mathbf{F}(k-1)\hat{\mathbf{x}}(k-1 \mid k-1) + \mathbf{G}(k-1)\mathbf{u}(k-1) \\
&\nu(k) = \mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k \mid k-1) \\
&\hat{\mathbf{x}}(k \mid k) = \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\nu(k).
\end{aligned}$$

---

that reference [5] sets $q = \sigma_a^2$, with the corresponding difference in the result. The insertion of f_{sc} is necessary in order to make the units consistent. Its interpretation is suggested by that of σ_a^2.

The quantity $\nu(k)$ is the *measurement residual* or *innovation*, $\mathbf{S}(k)$ is the *innovation covariance*, and $\mathbf{W}(k)$ is the *gain*.

The similarity to the toy problem of section 2.1 should now be apparent. In that problem we were dealing with a single quantity $\mathbf{x} = x$ to be estimated. That quantity was unchanging, so $\mathbf{F}(k) = 1$, $\mathbf{Q}(k) = 0$, and $\mathbf{u}(k) = 0$. We were measuring it directly, so $\mathbf{H}(k) = 1$, but there was some uncertainty in our measurements: $\mathbf{R}(k) = \sigma^2$. With minor changes in notation, equations 2.2-9 then become equivalent to equations 2.1-3.

One of the many virtues of the KF is that there is no need for the time between successive measurements to be constant. Hence, the fact that our sensors may not always detect the target (the probability of detection $P_D < 1$) is not in itself a problem for the validity of the filter, since missed measurements merely lead to greater gaps between successful measurements. If we wish to incorporate a no-measurement step into the KF, the innovation is treated as zero and the innovation covariance is not used to reduce the estimate covariance in that step. In other words,

$$\begin{aligned} \hat{\mathbf{x}}(k \mid k) &= \hat{\mathbf{x}}(k \mid k-1) \\ \mathbf{P}(k \mid k) &= \mathbf{P}(k \mid k-1) \end{aligned} \qquad \text{2.2-10}$$

when there is no measurement. (In effect, the gain is set to zero.)

An alternative formulation of the Kalman Filter, known as the *information filter* [6], is given by:

$$\begin{aligned} \mathbf{P}(k \mid k-1) &= \mathbf{F}(k-1)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^{\mathrm{T}}(k-1) + \mathbf{Q}(k-1) \\ \hat{\mathbf{x}}(k \mid k-1) &= \mathbf{F}(k-1)\hat{\mathbf{x}}(k-1 \mid k-1) + \mathbf{G}(k-1)\mathbf{u}(k-1) \\ \mathbf{P}^{-1}(k \mid k) &= \mathbf{P}^{-1}(k \mid k-1) + \mathbf{H}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k)\mathbf{H}(k) \\ \mathbf{P}^{-1}(k \mid k)\hat{\mathbf{x}}(k \mid k) &= \mathbf{P}^{-1}(k \mid k-1)\hat{\mathbf{x}}(k \mid k-1) + \mathbf{H}^{\mathrm{T}}(k)\mathbf{R}^{-1}(k)\mathbf{z}(k). \end{aligned} \qquad \text{2.2-11}$$

The term *information filter* arises from the fact that this formulation of the KF refers to the inverse of the covariance $\mathbf{P}$. The inverse covariance matrix can be interpreted as an amount of information available – more information meaning less uncertainty[5].

For justification of the Kalman Filter, see [6] or any of numerous other books or articles.

Notice that the KF depends heavily on the choice of process model and measurement model. A badly chosen model can lead to unacceptably poor performance. To some extent, one can compensate for a bad choice of model by increasing the process noise, but as a tradeoff this leads to larger uncertainty as represented by the $\mathbf{P}$ matrix, and damages the validity of the filter. In our aircraft tracking example, our process model will prove adequate if the aircraft moves with roughly constant velocity. If it accelerates steadily for a while, the KF based on our model will not keep up with the acceleration (for example, if the aircraft turns, our estimate of its position will be outside the turn), even if we try to compensate by increasing the process noise. Suppose instead that we extend the model to include the components of

[5] The track-level fusion techniques of section 7.3 will be seen to be analogous to the information filter formulation of the KF.

acceleration as part of the state vector, and make the acceleration, instead of the velocity, perturbed randomly by the process noise. This will handle periods of steady acceleration well, although transitions from one level of acceleration to another (*mode changes*) may not be handled well because those transitions generally do not fit the profile of Gaussian white noise. There are fancier algorithms available to handle these difficulties. Such algorithms (see section 2.5 for an example) are themselves based on the KF.

2.3 The Extended Kalman Filter

The KF assumes a linear process model and a linear measurement model, while there may be cases where a more realistic model is nonlinear. To be more general than in the last section, we may express the process and measurement models thus:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{f}[k, \mathbf{x}(k), \mathbf{u}(k)] + \mathbf{v}(k) \\ \mathbf{z}(k) &= \mathbf{h}[k, \mathbf{x}(k)] + \mathbf{w}(k). \end{aligned} \qquad 2.3\text{-}1$$

A technique called the *Extended Kalman Filter* (EKF) makes use of a Taylor series approximation to the process and measurement models. (The process model series is expanded around the current estimate, while the measurement model series is expanded around the next prediction. This calls for a new series expansion at each iteration.) The first-order EKF looks much like the regular KF, with a calculation of the effective transition and measurement matrices inserted:

$$\begin{aligned} \mathbf{F}(k-1) &= \left.\frac{\partial \mathbf{f}(k-1)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(k-1|k-1)} \\ \hat{\mathbf{x}}(k \mid k-1) &= \mathbf{f}[k-1, \hat{\mathbf{x}}(k-1 \mid k-1), \mathbf{u}(k-1)] \\ \mathbf{H}(k) &= \left.\frac{\partial \mathbf{h}(k)}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}(k|k-1)} \\ \mathbf{P}(k \mid k-1) &= \mathbf{F}(k-1)\mathbf{P}(k-1 \mid k-1)\mathbf{F}^{\mathrm{T}}(k-1) + \mathbf{Q}(k-1) \\ \mathbf{S}(k) &= \mathbf{H}(k)\mathbf{P}(k \mid k-1)\mathbf{H}^{\mathrm{T}}(k) + \mathbf{R}(k) \\ \mathbf{W}(k) &= \mathbf{P}(k \mid k-1)\mathbf{H}^{\mathrm{T}}(k)\mathbf{S}^{-1}(k) \\ \mathbf{P}(k \mid k) &= \mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^{\mathrm{T}}(k) \\ \nu(k) &= \mathbf{z}(k) - \mathbf{h}[k, \hat{\mathbf{x}}(k \mid k-1)] \\ \hat{\mathbf{x}}(k \mid k) &= \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\nu(k) \end{aligned} \qquad 2.3\text{-}2$$

where the notation $\mathbf{A} = \partial \mathbf{b} / \partial \mathbf{c}$ means that $A_{ij} = \partial b_i / \partial c_j$. Again this is based on [1].

When the EKF works, it works well; and this tends to be the case when the initial errors and the noises are small [1]. However, the EKF in general leads to prediction errors that are biased (i.e., have nonzero mean), and to inaccurate covariance matrices. Both of these problems can possibly result in very poor performance.

2.4 The Rectangular-Polar Problem

Measurements are likely to fall naturally into (spherical) polar coordinates[6]. The question arises as to what coordinate system should be used in the KF (or in whatever tracking algorithm we are using in its place). If we use polar coordinates for tracking, the process model will be nonlinear, while if we use cartesian coordinates, the measurement model will be nonlinear[7]:

$$\begin{bmatrix} r \\ \alpha \\ \theta \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan(z / \sqrt{x^2 + y^2}) \\ \arctan(y / x) \end{bmatrix} + \begin{bmatrix} w_r \\ w_\alpha \\ w_\theta \end{bmatrix} \qquad 2.4\text{-}1$$

(where r is the slant range, α is the elevation, and θ is the azimuth). Since we want to be able to share information from one platform to another, cartesian coordinates are more natural. The EKF of section 2.3 provides a possible way to deal with these nonlinearities. A better way (according to the studies in [1]) is described here. (We are assuming only position measurements, but presumably the technique could be modified for other types of kinematic measurements.)

A standard KF (or a KF modified for reasons other than the rectangular-polar problem) is used, with cartesian coordinates. The polar-coordinate measurement $\mathbf{z}(k)$ is transformed into an equivalent cartesian-coordinate measurement $\mathbf{z}^c(k)$, which is used in the filter. One might expect that the right transformation would be the inverse of equation 2.4-1 (without the noise term), namely

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\cos\alpha\cos\theta \\ r\cos\alpha\sin\theta \\ r\sin\alpha \end{bmatrix} \qquad 2.4\text{-}2$$

but in fact this leads to biases in the errors. That is, unbiased (zero mean error) polar measurements turn into biased (nonzero mean error) rectangular measurements. (This is a common problem with nonlinear transformations.) A de-biased transformation is achieved by simply subtracting the bias.

Assuming that the errors in the measured polar coordinates are zero-mean, Gaussian, and mutually uncorrelated, the de-biased transformation in three dimensions is:

---

[6] The spherical radial distance is often referred to as the *slant range*; the word *range* without a qualifier is used to refer to the cylindrical radial distance (i.e., the projection of the slant range onto the horizontal plane). The word *elevation* refers to the elevation angle, i.e., the angle between the sensor-target vector and the horizontal.

[7] Following common (but abysmal) practice, I am abusing the arctan function. By $\arctan(a/b)$ what is really meant here is $\begin{Bmatrix} \arctan(a/b), & b>0 \\ \arctan(a/b)\pm\pi, & b<0 \\ \pi/2, & b=0, a>0 \\ -\pi/2, & b=0, a<0 \end{Bmatrix}$, with an undefined result if $a=b=0$.

$$\mathbf{z}^c = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} r\cos\alpha\cos\theta(1-e^{-\sigma_\alpha^2-\sigma_\theta^2}+e^{-\sigma_\alpha^2/2-\sigma_\theta^2/2}) \\ r\cos\alpha\sin\theta(1-e^{-\sigma_\alpha^2-\sigma_\theta^2}+e^{-\sigma_\alpha^2/2-\sigma_\theta^2/2}) \\ r\sin\alpha(1-e^{-\sigma_\alpha^2}+e^{-\sigma_\alpha^2/2}) \end{bmatrix} \qquad 2.4\text{-}3$$

where r, α, and θ are the measured coordinates, and σ_α^2 and σ_θ^2 are the variances in α and θ respectively.

The de-biased transformed measurement $\mathbf{z}^c(k)$ is used in place of $\mathbf{z}(k)$ in the filter. Also, a covariance matrix $\mathbf{R}^c(k)$ corresponding to $\mathbf{z}^c(k)$ is used in place of $\mathbf{R}(k)$. Keeping the same assumptions about the errors in the measured coordinates, $\mathbf{R}^c(k)$ has the following components:

$$R_{xx}^c = e^{-2\sigma_\alpha^2-2\sigma_\theta^2}\begin{pmatrix} \cos^2\alpha\cos^2\theta\left[\left(r^2+2\sigma_r^2\right)\cosh 2\sigma_\alpha^2\cosh 2\sigma_\theta^2-\left(r^2+\sigma_r^2\right)\cosh\sigma_\alpha^2\cosh\sigma_\theta^2\right]+ \\ \cos^2\alpha\sin^2\theta\left[\left(r^2+2\sigma_r^2\right)\cosh 2\sigma_\alpha^2\sinh 2\sigma_\theta^2-\left(r^2+\sigma_r^2\right)\cosh\sigma_\alpha^2\sinh\sigma_\theta^2\right]+ \\ \sin^2\alpha\cos^2\theta\left[\left(r^2+2\sigma_r^2\right)\sinh 2\sigma_\alpha^2\cosh 2\sigma_\theta^2-\left(r^2+\sigma_r^2\right)\sinh\sigma_\alpha^2\cosh\sigma_\theta^2\right]+ \\ \sin^2\alpha\sin^2\theta\left[\left(r^2+2\sigma_r^2\right)\sinh 2\sigma_\alpha^2\sinh 2\sigma_\theta^2-\left(r^2+\sigma_r^2\right)\sinh\sigma_\alpha^2\sinh\sigma_\theta^2\right] \end{pmatrix}$$

$$R_{yy}^c = e^{-2\sigma_\alpha^2-2\sigma_\theta^2}\begin{pmatrix} \cos^2\alpha\cos^2\theta\left[\left(r^2+2\sigma_r^2\right)\cosh 2\sigma_\alpha^2\sinh 2\sigma_\theta^2-\left(r^2+\sigma_r^2\right)\cosh\sigma_\alpha^2\sinh\sigma_\theta^2\right]+ \\ \cos^2\alpha\sin^2\theta\left[\left(r^2+2\sigma_r^2\right)\cosh 2\sigma_\alpha^2\cosh 2\sigma_\theta^2-\left(r^2+\sigma_r^2\right)\cosh\sigma_\alpha^2\cosh\sigma_\theta^2\right]+ \\ \sin^2\alpha\cos^2\theta\left[\left(r^2+2\sigma_r^2\right)\sinh 2\sigma_\alpha^2\sinh 2\sigma_\theta^2-\left(r^2+\sigma_r^2\right)\sinh\sigma_\alpha^2\sinh\sigma_\theta^2\right]+ \\ \sin^2\alpha\sin^2\theta\left[\left(r^2+2\sigma_r^2\right)\sinh 2\sigma_\alpha^2\cosh 2\sigma_\theta^2-\left(r^2+\sigma_r^2\right)\sinh\sigma_\alpha^2\cosh\sigma_\alpha^2\right] \end{pmatrix}$$

$$R_{zz}^c = e^{-2\sigma_\alpha^2}\begin{pmatrix} \cos^2\alpha\left[\left(r^2+2\sigma_r^2\right)\sinh 2\sigma_\alpha^2-\left(r^2+\sigma_r^2\right)\sinh\sigma_\alpha^2\right]+ \\ \sin^2\alpha\left[\left(r^2+2\sigma_r^2\right)\cosh 2\sigma_\alpha^2-\left(r^2+\sigma_r^2\right)\cosh\sigma_\alpha^2\right] \end{pmatrix}$$

$$R_{xy}^c = R_{yx}^c = e^{-2\sigma_\alpha^2-4\sigma_\theta^2}\sin\theta\cos\theta\begin{pmatrix} \cos^2\alpha\left[\left(r^2+2\sigma_r^2\right)\cosh 2\sigma_\alpha^2-\left(r^2+\sigma_r^2\right)e^{2\sigma_\theta^2}\cosh\sigma_\alpha^2\right]+ \\ \sin^2\alpha\left[\left(r^2+2\sigma_r^2\right)\sinh 2\sigma_\alpha^2-\left(r^2+\sigma_r^2\right)e^{2\sigma_\theta^2}\sinh\sigma_\alpha^2\right] \end{pmatrix}$$

$$R_{xz}^c = R_{zx}^c = e^{-2\sigma_\alpha^2-\sigma_\theta^2/2}\sin\alpha\cos\alpha\cos\theta\left[\left(r^2+2\sigma_r^2\right)e^{-2\sigma_\alpha^2-\sigma_\theta^2/2}+\left(r^2+\sigma_r^2\right)\left(-e^{-\sigma_\alpha^2/2}-e^{-\sigma_\alpha^2/2-\sigma_\theta^2/2}\right)\right]$$

$$R_{yz}^c = R_{zy}^c = e^{-2\sigma_\alpha^2-\sigma_\theta^2/2}\sin\alpha\cos\alpha\sin\theta\left[\left(r^2+2\sigma_r^2\right)e^{-2\sigma_\alpha^2-\sigma_\theta^2/2}+\left(r^2+\sigma_r^2\right)\left(-e^{-\sigma_\alpha^2/2}-e^{-\sigma_\alpha^2/2-\sigma_\theta^2/2}\right)\right]$$

2.4-4

These three-dimensional de-biasing formulae are based on the two-dimensional ones presented in [1]. See Annex A for a partial derivation.

When this method is used to modify a plain KF, the resulting algorithm is known as the "Converted Measurement Kalman Filter, De-biased" (CMKF-D). Its more naïve counterpart, using 2.4-2 instead of 2.4-3 and a similarly simplified covariance matrix instead of 2.4-4, is the "Converted Measurement Kalman Filter, Linear" (CMKF-L).

2.5 The Interacting Multiple Model Filter

Sometimes the kind of process model that a KF (or EKF) requires is not adequately representative of the actual behaviour of the target of interest. In particular, a manoeuvring target may be better described by a discrete set of behaviour modes, with occasional switching from one mode to another. If each mode is representable by the kind of process model we have seen so far, then one of the possible ways of tracking the object is with the *Interacting Multiple Model* (IMM) filter.

We need a process model for each mode. The transition matrices $\mathbf{F}_\alpha(k)$ and process noise covariance matrices $\mathbf{Q}_\alpha(k)$ can be different for each mode. (The dimension of the state vector must be formally the same for each mode. Differing dimensions among modes can be handled with various tricks such as padding with zeroes, etc., but no more will be said about this matter.) In this discussion Greek indices will be used to label the modes.

Returning to the aircraft-tracking example from our discussion of the KF (but simplifying to two dimensions), the state vector is

$$\mathbf{x}(k) = \begin{bmatrix} x(k) & y(k) & \dot{x}(k) & \dot{y}(k) \end{bmatrix}^T. \qquad 2.5\text{-}1$$

Suppose we wish to model the aircraft's motion as an occasional switching among three modes, corresponding to roughly straight motion, hard turning to the left, and hard turning to the right. The respective transition matrices are

$$\mathbf{F}_1(k) = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$\mathbf{F}_2(k) = \begin{bmatrix} 1 & 0 & \frac{1}{\omega}\sin\omega T & -\frac{1}{\omega}(1-\cos\omega T) \\ 0 & 1 & \frac{1}{\omega}(1-\cos\omega T) & \frac{1}{\omega}\sin\omega T \\ 0 & 0 & \cos\omega T & -\sin\omega T \\ 0 & 0 & \sin\omega T & \cos\omega T \end{bmatrix}, \qquad 2.5\text{-}2$$

and $\mathbf{F}_3$ is the same as $\mathbf{F}_2$ except with ω replaced at each occurrence by $-\omega$. Here ω is whatever turning rate we wish to accuse our target of being capable of. (More realistically, the turning rate would be velocity-dependent. We have avoided this in order to keep the model linear.) We also have to decide on the process noise covariance for each of the three modes in order to complete the process model.

We also need, as part of the overall model, the *transition probabilities* $p_{\alpha\beta}$ (i.e., the estimated probability of the target being in mode β at time k given that it is in mode α at time $k-1$). These probabilities depend on the time between scans T and on the assumed *sojourn time* τ_α of a manoeuvre (i.e., the typical amount of time that we expect the target to stay in that manoeuvre). For the example above, if we imagine that our aircraft might spend

20 seconds on average in a turn, and the time between scans is fixed at 1 second, then we might use $p_{21} = p_{31} = \frac{1}{20}$, $p_{32} = p_{23} = 0$, $p_{22} = p_{33} = \frac{19}{20}$. If we also imagine that the aircraft might spend 30 seconds on average in roughly straight motion, and is as likely to turn left as to turn right, then we might use $p_{12} = p_{13} = \frac{1}{60}$, $p_{11} = \frac{29}{30}$. More generally, we have to account for the possible variability in T. Customarily, $p_{\alpha\alpha} = \max(1 - \frac{T}{\tau_\alpha}, l_\alpha)$ where l_α is a predetermined (arbitrary or intuitive) minimum value. The off-diagonal transition probabilities are related to each other according to some predetermined (arbitrary or intuitive) ratios, and calculated by applying the constraint $\sum_\beta p_{\alpha\beta} = 1$ for each α.

Recall that the KF kept at each stage the estimate $\hat{\mathbf{x}}(k \mid k)$ and its associated covariance $\mathbf{P}(k \mid k)$. The IMM keeps an estimate $\hat{\mathbf{x}}_\alpha(k \mid k)$ and associated covariance $\mathbf{P}_\alpha(k \mid k)$ for each mode, as well as a probability $\mu_\alpha(k)$ of the target being in mode α at time k. (Here $\alpha = 1,...,r$ where r is the number of modes, three in the example above).

The IMM algorithm runs as follows [1]. First, we calculate the mixed state estimates $\hat{\mathbf{x}}'_\alpha$ and mixed covariances $\mathbf{P}'_\alpha$, using the mixing coefficients $\mu_{\alpha\beta}$:

$$a_\beta(k-1) = \sum_{\alpha=1}^{r} p_{\alpha\beta}\mu_\alpha(k-1)$$

$$\mu_{\alpha\beta}(k-1) = \frac{1}{a_\beta(k-1)} p_{\alpha\beta}\mu_\alpha(k-1) \qquad 2.5\text{-}3$$

$$\hat{\mathbf{x}}'_\beta(k-1 \mid k-1) = \sum_{\alpha=1}^{r} \hat{\mathbf{x}}_\alpha(k-1 \mid k-1)\mu_{\alpha\beta}(k-1)$$

$$\mathbf{P}'_\beta(k-1 \mid k-1) = \sum_{\alpha=1}^{r} \mu_{\alpha\beta}(k-1)\Big(\mathbf{P}_\alpha(k-1 \mid k-1) + \left[\hat{\mathbf{x}}_\alpha(k-1 \mid k-1) - \hat{\mathbf{x}}'_\beta(k-1 \mid k-1)\right]\left[\hat{\mathbf{x}}_\alpha(k-1 \mid k-1) - \hat{\mathbf{x}}'_\beta(k-1 \mid k-1)\right]\Big)$$

Then, for each mode, $\hat{\mathbf{x}}_\alpha(k \mid k)$ and $\mathbf{P}_\alpha(k \mid k)$ are calculated via a standard KF (or EKF) using the measurement $\mathbf{z}(k)$ along with $\hat{\mathbf{x}}'_\alpha(k-1 \mid k-1)$ and $\mathbf{P}'_\alpha(k-1 \mid k-1)$ as inputs. (Notice the primes!) To complete the iteration, the updated mode probabilities are calculated by:

$$\Lambda_\alpha(k) = \frac{\exp\left[-\frac{1}{2}\nu_\alpha^T(k)\mathbf{S}_\alpha^{-1}(k)\nu_\alpha(k)\right]}{\sqrt{\det(2\pi\mathbf{S}_\alpha(k))}}$$

$$\mu_\alpha(k) = \frac{\Lambda_\alpha(k)a_\alpha(k-1)}{\sum_{\beta=1}^{r}\Lambda_\beta(k)a_\beta(k-1)} \qquad 2.5\text{-}4$$

where ν_α and $\mathbf{S}_\alpha$ are the innovation and innovation covariance in the α^{th} KF/EKF calculation.

Lastly, for purposes of output (and comparison with other algorithms), the overall state estimate and covariance are:

$$\hat{\mathbf{x}}(k \mid k) = \sum_{\alpha=1}^{r} \hat{\mathbf{x}}_\alpha(k \mid k)\mu_\alpha(k)$$

$$\mathbf{P}(k \mid k) = \sum_{\alpha=1}^{r} \mu_\alpha(k)\left(\mathbf{P}_\alpha(k \mid k) + \left[\hat{\mathbf{x}}_\alpha(k \mid k) - \hat{\mathbf{x}}(k \mid k)\right]\left[\hat{\mathbf{x}}_\alpha(k \mid k) - \hat{\mathbf{x}}(k \mid k)\right]^T\right) \quad \text{2.5-5}$$

Note that these last results are not used in the iteration.

2.6 Summary

In this chapter, we considered methods for tracking a single target in the absence of any questions of association.

The Kalman Filter (section 2.2) is a reliable, stable, method that can be derived rigorously from well-defined assumptions. It requires a linear process model and a linear measurement model.

The Extended Kalman Filter (section 2.3) is a natural extension of the Kalman Filter to nonlinear process and/or measurement models. In general it gives biased results, and it is very sensitive to the errors in the initial estimate.

The Converted Measurement Kalman Filter (section 2.4), which exists in both "Linear" and "De-biased" versions, is a more robust alternative to the Extended Kalman Filter for the case where the only nonlinearity in the models arises from measuring in polar coordinates while tracking in cartesian coordinates. The De-biased version has the added bonus of removing the bias that results from the nonlinear transformation.

The Interacting Multiple Model filter (section 2.5) is a Kalman Filter variant in which the process model can consist of several different modes of behaviour. It requires assumptions about the probabilities with which the target switches among the modes.

3. False Alarms

We have been assuming thus far that we have no difficulty with association; i.e., that we have a track that belongs (we are sure) to something real, and we occasionally have a new measurement that is known to be of the same target. In a real tracking situation, there is in general a nonzero probability of our sensor picking up spurious signals[8], which can be mistakenly associated with a target of interest. That is, we have a probability of false alarm $P_{FA} > 0$. (The response of a sensor when examining a given region of space is usually compared to a threshold, and by adjustment of the threshold we can have some control over P_{FA}. However, a decrease in P_{FA} then goes hand in hand with a decrease in P_D, the probability of detection[9].) Thus we may, at any given scan time, have multiple contacts where at most one of them belongs properly to the target of interest. So even with only one target, we have the problem of association; that is, we have to decide which contact (if any) belongs to the target.

3.1 Gates and Validation

In many cases it will be obvious that a given contact does not belong to the target of interest – for example, if the new contact is in the wrong direction altogether. In order to cut down on the number of spurious contacts, it is customary to impose a *gate*; that is, a threshold on the difference between the expected target state and the apparent contact state. A traditional validation criterion for the measurement $\mathbf{z}(k)$ is

$$\nu^T(k)\mathbf{S}^{-1}(k)\nu(k) \le \gamma \qquad 3.1\text{-}1$$

where γ, the *gate threshold*, can be thought of as the square of the "number of standard deviations" around the expected state that we allow the measured state to possibly fall into. Measurements that fall outside the gate are discarded, while those that fall inside the gate may be used to update the track. Note that the innovation covariance $\mathbf{S}(k)$ must be calculated before judging whether a given measurement is inside the gate.

The gate threshold should be chosen to give a high probability P_G of the true measurement (if there is one) falling into the gate; preferably 99% or more. It is customary to assume that the innovation (from a measurement of the actual target) will be normally distributed about zero with covariance $\mathbf{S}(k)$. (This follows from the assumptions behind the KF process model.) Under this assumption, the gate probability P_{G_m} for a standard gate in an m-dimensional measurement space is given by a chi-squared distribution with m degrees of freedom. Chi-squared values can be looked up in statistical tables. The 99% gate threshold is 6.635 for one dimension, 9.210 for two dimensions, and 11.345 for three dimensions.

---

[8] One often sees the word *clutter* used in connection with the problem of false alarms. A formal definition seems to be hard to find. From the contexts in which the word appears, it appears to be variously defined as (1) false alarms considered collectively, (2) a high concentration of false alarms, or (3) the (unspecified) cause of false alarms.

[9] P_D is usually defined as a probability per scan, while P_{FA} is usually defined as a probability per resolution cell per scan.

The kind of gate described above assumes that we are using a single mode KF/EKF for tracking. If we are using a multiple-mode method such as an IMM filter, there will be a different innovation $\nu(k)$ and a different innovation covariance $\mathbf{S}(k)$ for each KF in the combined model. In this case the specification of the gate presents a peculiar problem. One option is to construct the gate from the union of the individual gates derived from the sub-models. In other words, the measurement is considered to fall in the gate if

$$\nu_\alpha^{\mathrm{T}}(k)\mathbf{S}_\alpha^{-1}(k)\nu_\alpha(k) \le \gamma \qquad 3.1\text{-}2$$

for at least one mode α. However, keeping in mind that the primary purpose of a gate is to reduce the computational load on your data fusion system, it may be preferable to use a gate with a simpler definition, even if that gate is larger than it needs to be. If the sub-models of your IMM differ only in their process noise (which, according to [5], can be useful provided that the noise values all differ in order of magnitude), then it suffices to use the single largest gate, i.e., the gate derived from the sub-model with the largest process noise. But if the sub-models differ in other ways (as in the example given in section 2.5) then a simple gate definition (based on a single sub-model) may require a very large value of γ in order to ensure a 99% (or more) probability of the true measurement falling into the gate, independently of which mode best describes the target's current behaviour. In all these cases, a calculation of the exact gate probability may be more difficult than it is worth. Algorithms that make explicit use of the gate probability will not usually suffer much from making the approximation $P_G = 1$.

Some algorithms make use of the volume of the validation (gate) region, which for a standard gate is:

$$V(k) = c_m\sqrt{\gamma^m \det \mathbf{S}(k)} \qquad 3.1\text{-}3$$

where

$$c_m = \frac{\pi^{m/2}}{\Gamma(\frac{m}{2}+1)}. \qquad 3.1\text{-}4$$

We have already alluded to the importance of appropriate process and measurement models when using a KF or EKF. Finding an appropriate model is in general a nontrivial problem. Any deviation from what your model "expects" may lead to a valid measurement falling outside the gate and being mistakenly thrown out as a false alarm – this will probably lead to loss of the track. Such track loss can be made less likely by increasing the gate threshold γ or increasing the process noise (which causes the components of $\mathbf{S}(k)$ to increase), but the side effect is an increase in the number of false alarms that fall inside the gate. There is therefore a tradeoff between catching manoeuvres and dealing with clutter. The remainder of this chapter is concerned with methods for dealing with false alarms. We will assume that there is only one target of interest, and that false alarms are uniformly distributed over all space.

3.2 Nearest-Neighbour and Strongest-Neighbour Methods

The simplest method is the *Nearest Neighbour* (NN) method[10], which consists simply of selecting the nearest measurement and discarding the rest, as long as at least one measurement is validated (i.e., falls into the gate). "Nearest," in this case, means having the smallest value of $\nu^{\mathrm{T}}(k)\mathbf{S}^{-1}(k)\nu(k)$, for the measurements at time k. The selected measurement is used, via a KF or other single-target filter, to update the track.

As a variation on this method, if signal intensity information is available, one could pick the strongest measurement that falls in the gate instead of the nearest one; this is the *Strongest Neighbour* (SN) method.

3.3 The Probabilistic Data Association Filter

The *Probabilistic Data Association Filter* (PDAF) is an "all-neighbours" algorithm. That is, all the validated measurements are used to update the state estimate. We are still assuming that we have only one target of interest, but now we are allowing for false alarms as well. With respect to the false alarms, the assumption is that they appear randomly throughout the surveillance area. If we know the spatial density λ of false alarms, the algorithm can make use of this, but it is not necessary. The performance of the algorithm may be degraded by a persistent false alarm, for example from a real object in the vicinity of the target of interest.

3.3.1 Ordinary PDAF

As in the standard KF, at each stage we keep the current estimate $\hat{\mathbf{x}}(k \mid k)$ and its associated covariance $\mathbf{P}(k \mid k)$. In addition to the usual model parameters $\mathbf{F}(k)$, $\mathbf{H}(k)$, $\mathbf{Q}(k)$, and $\mathbf{R}(k)$, we also need to know (or guess) P_{D}, the prior probability that our target of interest will be detected at each iteration. Ideally, we should also know P_{G}, the gate probability as calculated above, but it will have little impact on the filter if we make the approximation $P_{\mathrm{G}} = 1$.

The algorithm runs as follows [1]. Starting from $\hat{\mathbf{x}}(k-1 \mid k-1)$ and $\mathbf{P}(k-1 \mid k-1)$, we calculate $\hat{\mathbf{x}}(k \mid k-1)$, $\mathbf{P}(k \mid k-1)$, $\mathbf{S}(k)$, and $\mathbf{W}(k)$ as in a standard KF (or EKF). Having $\mathbf{S}(k)$ allows us to define a gate for validation of the measurements. Let $M(k)$ denote the number of validated measurements, and let $\mathbf{z}_i(k)$ denote the values of those measurements. Associated with each measurement is the innovation $\nu_i(k)$, calculated from $\mathbf{z}_i(k)$ in the usual way.

As advertised, the PDAF makes use of a weighted average of the (validated) innovations. We need to calculate the weighting factors, denoted β_i, $i = 0,\ldots, M(k)$. β_0 is the probability that none of the validated measurements belong to the target, while for $i = 1,\ldots, M(k)$, β_i is the probability that the i^{th} measurement belongs to the target. They are given by:

---

[10] This term is also used for the analogous multi-target association method; see section 5.1.

$$\beta_0 = \frac{b}{b + \sum_{j=1}^{M(k)} e_j}$$

$$\beta_i = \frac{e_i}{b + \sum_{j=1}^{M(k)} e_j} \Leftarrow i = 1,\ldots, M(k) \qquad \text{3.3.1-1}$$

where

$$e_i = \exp\left[-\tfrac{1}{2} \nu_i(k)^{\mathrm{T}} \mathbf{S}^{-1}(k) \nu_i(k)\right]$$

$$b = \left(\frac{2\pi}{\gamma}\right)^{\frac{m}{2}} \frac{M(k)}{c_m}\left(\frac{1}{P_{\mathrm{D}}} - P_{\mathrm{G}}\right) \qquad \text{3.3.1-2}$$

This formula for b is for the so-called *nonparametric PDAF*, which is based on equal prior probabilities of every possible number of false alarms. It makes more theoretical sense, if the false alarm density λ is known, to assume a prior Poisson distribution for the number of false alarms (probability of N_{FA} false alarms being $e^{-\lambda V}(\lambda V)^{N_{\mathrm{FA}}} / N_{\mathrm{FA}}!$). For this *parametric PDAF*, we replace $M(k)$ with $\lambda V(k)$ in the formula for b. According to [1], there seems to be little evidence of significant improvement in performance when this substitution is made, however.

Having worked out the weighting factors, we finish off the iteration thus:

$$\begin{aligned}
\nu(k) &= \sum_{i=1}^{M(k)} \beta_i(k) \nu_i(k) \\
\hat{\mathbf{x}}(k \mid k) &= \hat{\mathbf{x}}(k \mid k-1) + \mathbf{W}(k)\nu(k) \\
\bar{\mathbf{P}}(k) &= \mathbf{P}(k \mid k-1) - \mathbf{W}(k)\mathbf{S}(k)\mathbf{W}^{\mathrm{T}}(k) \qquad \text{3.3.1-3} \\
\tilde{\mathbf{P}}(k) &= \mathbf{W}(k)\left[\sum_{i=1}^{M(k)} \beta_i(k)\nu_i(k)\nu_i(k)^{\mathrm{T}} - \nu(k)\nu^{\mathrm{T}}(k)\right]\mathbf{W}^{\mathrm{T}}(k) \\
\mathbf{P}(k \mid k) &= \beta_0(k)\mathbf{P}(k \mid k-1) + \left[1 - \beta_0(k)\right]\bar{\mathbf{P}}(k) + \tilde{\mathbf{P}}(k).
\end{aligned}$$

Here $\bar{\mathbf{P}}(k)$ represents the value that $\mathbf{P}(k \mid k)$ would have if we were perfectly confident about which measurement was the correct one, while $\tilde{\mathbf{P}}(k)$ represents the increase in the state estimate covariance due to the uncertainty of association.

In general the performance of the PDAF is far superior to that of the NN method, if the latter is combined with the same KF (or EKF) that is used for the filtering aspect of the PDAF.

3.3.2 IMMPDAF

As mentioned previously, there is a conflict between handling clutter and handling manoeuvers. The description of the PDAF above is based on a standard KF (or EKF), but it is possible to base the PDAF on an IMM filter, thus combining the clutter-handling capability of the former with the manoeuvre-handling capability of the latter. This method is (perhaps not surprisingly) called the *Interacting Multiple Model Probabilistic Data Association Filter*, or IMMPDAF.

As in an ordinary IMM, the IMMPDAF requires the measurement model parameters $\mathbf{H}(k)$ and $\mathbf{R}(k)$, the mode-specific process model parameters $\mathbf{F}_\alpha(k)$ and $\mathbf{Q}_\alpha(k)$, and the mode transition probabilities $p_{\alpha\beta}$. As in an ordinary PDAF, we also need the detection probability P_D. As for the gate probability, one might as well use the approximation $P_G = 1$ here. In presenting the algorithm, Greek indices will be used for the modes while Roman indices will be used for the measurements and the corresponding association hypotheses.

This presentation of the algorithm is based on clues scattered through [1]. Starting from the mode-dependent state estimate $\hat{\mathbf{x}}_\alpha(k-1|k-1)$, its corresponding covariance $\mathbf{P}_\alpha(k-1|k-1)$, and the mode probability $\mu_\alpha(k-1)$, we calculate the mixing coefficients $a_\alpha(k-1)$ and $\mu_{\alpha\beta}(k-1)$, the mixed state estimate $\hat{\mathbf{x}}'_\alpha(k-1|k-1)$, the mixed covariance $\mathbf{P}'_\alpha(k-1|k-1)$, the predictions $\hat{\mathbf{x}}_\alpha(k|k-1)$ and $\mathbf{P}_\alpha(k|k-1)$, the innovation covariance $\mathbf{S}_\alpha(k)$, and the gain $\mathbf{W}_\alpha(k)$ as in the ordinary IMM filter.

Define a gate for validation of the measurements using the innovation covariances (see the description of multiple-mode gates in section 3.1). Let $M(k)$ be the number of validated measurements. For each validated measurement $\mathbf{z}_i(k)$, we calculate the mode-dependent innovations

$$\boldsymbol{\nu}_{\alpha i}(k) = \mathbf{z}_i(k) - \mathbf{H}(k)\hat{\mathbf{x}}_\alpha(k|k-1). \qquad 3.3.2\text{-}1$$

The mode-dependent weighting factors are

$$\beta_{\alpha 0} = \frac{b_\alpha}{b_\alpha + \sum_{j=1}^{M(k)} e_{\alpha j}}$$

$$\beta_{\alpha i} = \frac{e_{\alpha i}}{b_\alpha + \sum_{j=1}^{M(k)} e_{\alpha j}} \Leftarrow i = 1,\ldots,M(k) \qquad 3.3.2\text{-}2$$

where

$$e_{\alpha i} = \exp\left[-\tfrac{1}{2}\boldsymbol{\nu}_{\alpha i}^{\mathrm{T}}(k)\mathbf{S}_\alpha^{-1}(k)\boldsymbol{\nu}_{\alpha i}(k)\right]$$

$$b_\alpha = (2\pi)^{\frac{m}{2}}\left(\frac{1}{P_D} - 1\right)\frac{M(k)}{V(k)}\sqrt{\det(\mathbf{S}_\alpha(k))}. \qquad 3.3.2\text{-}3$$

In this last expression, which is appropriate for a nonparametric PDAF, $V(k)$ is the volume of the validation gate. If the gate is based on a single representative mode Γ, i.e., if we are using the gate criterion

$$\boldsymbol{\nu}_{\Gamma j}(k)^{\mathrm{T}}\mathbf{S}_\Gamma^{-1}(k)\boldsymbol{\nu}_{\Gamma j}(k) \le \gamma, \qquad 3.3.2\text{-}4$$

then

$$V(k) = c_m\sqrt{\gamma^m \det\mathbf{S}_\Gamma(k)} \qquad 3.3.2\text{-}5$$

so the expression for b_α becomes

$$b_\alpha = \left(\frac{2\pi}{\gamma}\right)^{\frac{m}{2}} \left(\frac{1}{P_D} - 1\right) \frac{M(k)}{c_m} \sqrt{\frac{\det(\mathbf{S}_\alpha(k))}{\det(\mathbf{S}_T(k))}}. \qquad 3.3.2\text{-}6$$

If our gate is constructed in a way that makes its volume difficult to calculate, then an alternative is to let $V(k)$ be the volume of the entire surveillance region, and replace $M(k)$, where it appears in the formula for b_α, by the total number of measurements (validated or not) at time k. We may still ignore the non-validated measurements in the summations in the formulae for the $\beta_{\alpha i}$, since these measurements will contribute very little to those sums. An even better alternative, if the false alarm density λ is known, is to use the parametric PDAF, i.e., replace the ratio $\frac{M(k)}{V(k)}$ by λ.

Interpret $\beta_{\alpha 0}$ as the probability that none of the measurements is from the target, and $\beta_{\alpha i}$ as the probability that measurement $\mathbf{z}_i$ is from the target, under the assumption that mode α is a good description of the target's behaviour.

The combined mode-dependent innovation $\nu_\alpha(k)$, the updated mode-dependent state estimate $\hat{\mathbf{x}}_\alpha(k \mid k)$, and its covariance $\mathbf{P}_\alpha(k \mid k)$ are also worked out as in the ordinary PDAF (i.e., by equations 3.3.1-3, with a mode index attached to each variable). The likelihood function $\Lambda_\alpha(k)$ is given by

$$\Lambda_\alpha(k) = \frac{1 - P_D}{V(k)^{M(k)}} + \frac{P_D \sum_{i=1}^{M(k)} \exp\left[-\frac{1}{2}\hat{\mathbf{i}}_{\alpha i}^{\mathrm{T}}(k)\mathbf{S}_\alpha^{-1}(k)\hat{\mathbf{i}}_{\alpha i}(k)\right]}{M(k)V(k)^{M(k)-1}\sqrt{\det(2\pi\mathbf{S}_\alpha(k))}} \qquad 3.3.2\text{-}7$$

and the updated mode probability $\mu_\alpha(k)$ is worked out as in an ordinary IMM filter. Again, if our gate has a complicated size and shape, it may be necessary to use the entire surveillance region for $V(k)$, in which case we must replace $M(k)$ by the total number of measurements (validated or not), except in the summation (which can remain unmodified). Also, we can again replace $M(k)$ by $\lambda V(k)$ if the false alarm density λ is known. From the formula for $\mu_\alpha(k)$ (equation 2.5-4), one can see that the likelihood functions can be multiplied by any constant without affecting the result. Thus we can get rid of most of the factors of $V(k)$ in any case, and can get rid of all of the volume factors if λ is used.

3.4 Summary

In this chapter, we considered methods for deciding whether to treat a contact as a real target or a false alarm.

The Nearest-Neighbour method (section 3.2) is a simple association algorithm that combines easily with any tracking algorithm. The Strongest-Neighbour method (also in section 3.2) is also simple, but requires that the measurements include signal intensity data.

The Probabilistic Data Association Filter (PDAF, section 3.3) is a more sophisticated algorithm, combining the roles of an association algorithm and a tracking algorithm. Because these roles are intertwined, it is much less straightforward to base the PDAF on any given tracking algorithm than to combine the Nearest-Neighbour method with that same tracking algorithm. Two kinds of PDAF were presented in this chapter, one based on the Kalman Filter and the other based on the Interacting Multiple Model filter. The former version of the PDAF is known to give better results than the combination of a Kalman Filter with Nearest-Neighbour association.

4. Track Initiation and Deletion

In Chapters 2 and 3, we assumed that we had a track that was already initiated, and we were concerned only with using further incoming data to update the track. In Chapter 5 we will consider similar questions for the case of multiple targets. We now address the question of how tracks are formed in the first place.

Note that methods described here assume the possibility of multiple tracks, even though we have not yet discussed the maintenance of multiple tracks. The choice of initiation procedure depends to some extent on the choice of procedure for maintaining multiple tracks. The *Multiple-Hypothesis Tracking* (MHT) method, described in section 5.3, is often said to have its own track initiation built in, thus rendering any separate track-initiation procedure (such as "2/2&m/n") unnecessary. Nevertheless, comments on the MHT will be made at the end of this chapter.

This chapter is organized around the methods for initiating tracks. A good MSDF system must also have the capability to remove tracks, which is a much simpler matter. Brief comments on track deletion will also appear in the following sections.

4.1 "2/2 & m/n"

Now we will describe the "2/2 & m/n" method. (Again this is based on [1].) The numbers m and n (not to be confused with the m and n used previously for numbers of dimensions) are parameters that must be chosen; typical choices for m/n in radar applications are 1/2, 2/3, 2/4, 3/4, or 3/5. Where sonar is being used, m and n are often much higher (e.g., 5/10).

As mentioned in the introduction, a track may include a parameter that rates the degree of certainty that the track represents something real. For now we will consider this parameter to have three possible discrete values, *tentative, preliminary,* and *confirmed.* A confirmed track is one that is strongly believed to belong to something real, so the discussions of track maintenance in chapters 2 and 3 apply.

In the "2/2 & m/n" method, every contact that is not used to update an existing track is used to form a *tentative track*. This will include all contacts that do not fall into the gate of an existing track, and if an NN-type algorithm is being used, it may even include some contacts that do fall into such a gate. A tentative track is thus derived from a single contact.

Some of the properties of the track will be left blank; for example, if we are measuring the position only, then the velocity will be utterly unknown at this time. The gate for a tentative track must be large enough to accommodate this uncertainty. Let $(\Delta z_i)_{\max}$ denote the maximum change in the i^{th} measured component of the target kinematics from one measurement to the next. More precisely, it denotes the maximum value of the i^{th} component of $\mathbf{H}(\mathbf{x}(k) - \mathbf{x}(k-1))$. For example, if we are measuring position in two dimensions, then $(\Delta z_1)_{\max} = (\Delta z_2)_{\max} = v_{\max} T$ where $v_{\max}$ is the maximum speed of the target (use the maximum speed of all known target types if the first measurement did not give

us sufficient target type information to judge what its maximum speed should be) and, as usual, T is the time between measurements. Then a convenient gate condition at time k for a tentative track formed at time $k-1$ is

$$|z_i(k) - z_i(k-1)| \le (\Delta z_i)_{\max} + g_1\sqrt{2R_{ii}} \quad : \quad i = 1,\ldots,m \qquad 4.1\text{-}1$$

where $z_i(k)$ is the i^{th} component of the measurement $\mathbf{z}(k)$, R_{ii} is the i^{th} diagonal component of $\mathbf{R}$ (the covariance matrix of the measurement noise), and g_1 is the "number of standard deviations" that we wish to allow around the value that is expected if $(\Delta z_i)_{\max}$ is achieved by the target. ($g_1 = 2.5$ should be sufficient.)

Any tentative track that does not receive a contact in its gate at the first opportunity (i.e., in the very next scan or frame after the formation of the tentative track) is discarded. Also, if all contacts that fall into the tentative track's gate are used to update existing preliminary or confirmed tracks, then the tentative track is discarded.

Now suppose that we have a tentative track, that one contact falls into its gate at the first opportunity, and that this contact is not used to update a preliminary or confirmed track. The tentative track is then upgraded to a preliminary track, and the state estimate and covariance matrix used in the filtering algorithm must be initialized. This initialization can be done easily if the highest order of the kinematic components in the state vector in our process model is one higher than the highest order that is directly measured. For example, if we measure positions only, and the process model has a state vector with positions and velocities but not accelerations (as in the aircraft tracking example of section 2.2), then the two position measurements $\mathbf{z}(k-1)$ and $\mathbf{z}(k)$ will be enough to initialize a Kalman Filter:

$$\hat{\mathbf{x}}(k \mid k) = \begin{bmatrix} \mathbf{z}(k) \\ \frac{1}{T}(\mathbf{z}(k) - \mathbf{z}(k-1)) \end{bmatrix}$$

$$\mathbf{P}(k \mid k) = \begin{bmatrix} \mathbf{R}(k) & \frac{1}{T}\mathbf{R}(k) \\ \frac{1}{T}\mathbf{R}(k) & \frac{1}{T^2}(\mathbf{R}(k) + \mathbf{R}(k-1)) \end{bmatrix} \qquad 4.1\text{-}2$$

(Note that these equations are only for initialization of a preliminary track at time k; they are not recursive!) If, however, our process model includes acceleration, then two measurements are not enough (strictly speaking) to initialize the filter. Perhaps, in this case, we would assign a value of zero to the initial acceleration, with the corresponding uncertainty (in the $\mathbf{P}$ matrix) being set to (the square of) a large fraction of the maximum possible acceleration of which we can credit our target with being capable. (The additional off-diagonal components could be set to zero in this case.)

If more than one contact falls into the gate of a tentative track (again, without being used to update a preliminary or confirmed track), then a new preliminary track can be formed for each relevant combination.

A preliminary track is treated the same as a confirmed track, with respect to gating and updating, but its progress is monitored so that it will soon be either discarded or upgraded to a confirmed track. This is where the parameters m and n come in: If the preliminary track gets updated by a contact in at least m of the n scans immediately following the initialization of the

preliminary track, then it is upgraded to a confirmed track. This upgrading occurs immediately upon receiving the m^{th} hit, if it is not discarded first. A preliminary track is discarded immediately upon the $(n-m+1)^{th}$ scan in which it fails to get updated by a received contact, if it is not upgraded to a confirmed track first. The decision to upgrade or discard will therefore be made in at most n scans after formation of the preliminary track. (The track data structure needs to have room for the number of hits and misses since preliminary status was reached.) This is where the name "2/2 & m/n" comes from; the "2/2" refers to the two measurements that are needed to form the preliminary track, while the "m/n" refers to the m measurements that are needed to upgrade the track status to confirmed.

In the case where two measurements are not sufficient to properly initiate the filter (e.g., if the state vectors in our process model contain acceleration, but we measure position only), then an alternative to the arbitrary zero-acceleration initiation suggested above would be a "3/3 & m/n" initiation method. The gate for a newly-initiated tentative track would be the same as above, while the gate for a two-measurement tentative track would be constructed in an analogous way, but based on a "known" velocity (derived from the two positions) and a maximum possible acceleration. If the third measurement arrives on time, then the track would be upgraded to preliminary, and the filter initiated properly. Because it requires three measurements in a row to form a preliminary track, this method is stricter than the "2/2 & m/n" method and will therefore result in lower incidence of preliminary and confirmed tracks.

In choosing the parameters m and n, there are tradeoffs among various measures of performance: the probability of forming a false confirmed track, the average length (in number of scans) of a false preliminary track, the average time it takes to confirm a true track, and the probability of confirming a true track by a given number of scans. See [1] for further details on choosing these parameters.

A confirmed track that has not been updated for a sufficiently long time is no longer useful and ought to be deleted. We could use, for example, a maintenance condition of the form "m_d out of n_d" (with no necessary relationship between m_d and m or between n_d and n). In this case, if there are fewer than m_d successful measurements in any n_d consecutive attempts since the track was confirmed, the track is deleted. If m_d is set to one, this criterion is equivalent to saying that the track will be deleted if n_d misses occur in a row.

4.2 Track Initiation in the IMMPDAF

The IMMPDAF method of subsection 3.3.2 lends itself naturally to track formation with minor modifications [1]. Roughly, the trick is to include an "invisible" mode among the modes from which the IMM is constructed. The "invisible" mode has the same process model as the "roughly constant velocity" (non-manoeuvring) mode, but is modified by the assumption that the alleged target will be undetectable if it is in the "invisible" mode. Since an IMM filter keeps track of the probabilities of the target being in each mode, the probability of the target not being in the "invisible" mode can be considered a measure of the quality of the track.

The method runs as follows. Tentative and preliminary tracks are formed as in the "2/2&m/n" method (i.e., the "2/2" part). For each new preliminary track, an IMMPDAF is initialized (with the same initial state estimate and covariance for each sub-model). Let the 1^{st} mode be

the "invisible" mode and the 2nd mode be the "roughly constant velocity" mode; $\mathbf{F}_1 = \mathbf{F}_2$ and $\mathbf{Q}_1 = \mathbf{Q}_2$. There should be some small transition probabilities between these two modes. (We can proceed with only these two modes, in which case $p_{12} = p_{21} = 0.02$ and $p_{11} = p_{22} = 0.98$ are suitable.) The initial mode probabilities are $\mu_1 = \mu_2 = 0.5$.

Where track updating is concerned, all the modes except the "invisible" mode use the usual IMMPDAF formulae. For the "invisible" mode, the weighting factors are $\beta_{i_1} = 0$ for $i > 0$, and $\beta_{i0} = 1$, so that

$$\begin{aligned} \hat{\mathbf{x}}_1(k \mid k) &= \hat{\mathbf{x}}_1(k \mid k-1) \\ \mathbf{P}_1(k \mid k) &= \mathbf{P}_1(k \mid k-1). \end{aligned} \qquad 4.2\text{-}1$$

The detection probability is treated as zero in the "invisible" mode, so the likelihood function is simply

$$\Lambda_1 = \frac{1}{V(k)^{M(k)}}. \qquad 4.2\text{-}2$$

The fact that $\Lambda_1 \neq \Lambda_2$ (despite those two modes having the same process model) will cause the mode probabilities to change with time. The *true track probability* $1 - \mu_1(k)$ provides a measure of the credibility of the track. The track can be upgraded from "preliminary" to "confirmed" if the true track probability exceeds some predetermined threshold, such as 0.6. Upon being upgraded, the IMMPDAF can be expanded to include a larger number of modes, if desired. This method is sometimes used with only the two modes (invisible and roughly constant velocity) for preliminary tracks, with a variety of manoeuvring modes appended for confirmed tracks.

Track deletion is simply handled by watching to see if the true track probability goes below some other threshold, typically 0.05 (regardless of whether the track ever became confirmed).

4.3 Single-Measurement Track Initiation

It has been noted (in the opening of this chapter) that the MHT method of section 5.3 has its own track initiation built in. Because every possible hypothesis is considered, the *decision-making* aspect of track initiation is automatic; every measurement always has a possibility of belonging to a new target. However, in order to avoid unduly complicating the algorithm, it is desirable to treat a new single-measurement track as if it is fully formed. Thus, it is worthwhile commenting on how a state vector and its covariance might be determined (to abuse the word!) from a single measurement.

A hint of this has already been given in section 4.1, where we have seen that even two measurements may not be enough to form a track "properly", depending on the dimension of our state vector. In essence, every part of the state vector that can be matched to a part of the measurement is considered to be equal to that measurement (transformed as necessary), and the corresponding part of the covariance is simply the measurement uncertainty. Every part of the state vector that cannot be matched to a part of the measurement is given some central value (typically zero) and the corresponding part of the covariance is made very large in order

to cover the entire range of possibilities. For example, if our state vector consists of positions, velocities, and accelerations in cartesian coordinates, while our measurement is only of position, then the new track is assigned a position equal to the measured position, and a velocity and acceleration of zero. The elements of the covariance matrix corresponding to uncertainty in position are given values equal to the measurement covariance, those corresponding to uncertainty in velocity and acceleration are given values in the order of the square of the maximum possible velocity and acceleration of our target. The remaining off-diagonal elements can be set to zero.

It can easily be shown that (in principle) this procedure leads, after updating the track with further measurements, to a state vector with (close to) the "correct" state vectors and covariances. However, this is a point where particular care must be taken to ensure numerical stability (e.g., to avoid rounding errors).

4.4 Summary

In this chapter, we considered methods for the formation of new tracks.

The "2/2 & m/n" method (section 4.1) is a fairly straightforward track formation method that combines easily with any tracking and association algorithms.

The "invisible mode" method (section 4.2) is narrowly applicable, combining only with the Interacting Multiple Model Probabilistic Data Association Filter association-and-tracking algorithm. It is more sophisticated than the "2/2 & m/n" method.

Section 4.3 described the procedure of creating a fully-formed track from a single measurement. No methods for determining whether such a new track should be retained were presented.

5. Multiple Targets

We have already seen (in the chapter on false alarms) the problem of association and some ways of dealing with it. Association becomes a deeper problem when there are multiple targets. Not only do we need to decide which contacts are probably false, but we need to decide which probably-true contacts should go with which tracks. And, as before, the need to maintain tracks despite target manoeuvers tends to conflict with the need to associate data correctly.

For the sake of completeness, it should be mentioned that the choice of origin to which a given contact should be attributed is not always as simple as "this target, that target, ..., or false alarm". When two targets are in very nearly the same direction with respect to the sensor, there is the possibility that a contact may be an unresolved sensor return from both targets at once. This document will not deal with any explicit algorithmic modifications that are designed for the purpose of dealing with such unresolved hits.

5.1 Nearest-Neighbour Methods

The nearest neighbour (NN) methods described here will be of the "immediate decision" variety. That is, after each scan a firm decision is reached as to which contacts will be considered to belong to which tracks. Those tracks can then be updated using whichever single-target tracking method is desired. (In this section we will be assuming an ordinary Kalman Filter.)

In order to reach our association decision, we need a measure of "distance" for each contact-track pair. The "distance" between the i^{th} track and the j^{th} contact is denoted d_{ij}. The basic assumptions are [2] that (1) at most one track should be associated with each contact, (2) at most one contact should be associated with each track, (3) a given track and contact may only be associated if the contact falls into the gate (as in section 3.1) for that track, (4) the optimal association decision is one that maximizes the number of track-contact associations that are made within the constraints of assumptions 1 to 3, and (5) the optimal association decision is one that minimizes the summed "distances" in the associations made within the constraints of assumptions 1 to 4. Having reached an association decision, any unassociated contacts can be turned into tentative tracks (as in section 4.1).

The "distance" for a contact-track pair is based on a comparison of the predicted state $\hat{\mathbf{x}}_i(k \mid k-1)$ of the track in question with the measurement $\mathbf{z}_j(k)$. In general it will depend on which single-target tracking method is being used. For an ordinary Kalman Filter the distance is typically [2] taken to be

$$d_{ij} = \nu_{ij}^{\mathrm{T}}(k)\mathbf{S}_i^{-1}(k)\nu_{ij}(k) + \ln\left(\det \mathbf{S}_i(k)\right) \qquad 5.1\text{-}1$$

where $\nu_{ij}(k) = \mathbf{z}_j(k) - \mathbf{H}(k)\hat{\mathbf{x}}_i(k \mid k-1)$ is the usual innovation based on this contact-track pair. More complicated distance functions are appropriate for other tracking methods, but we will not deal with those complications here.

In order to aid in the association decision-making, it is convenient to construct an "association matrix" with a row for each track, and a column for each contact. The entries of the matrix are the values of the "distance" function, if the contact-track pair in question is validated (i.e., if the contact falls into the gate of the track), and is otherwise an infinity or other special symbol to indicate that the association in question is unfeasible.

Here is an example of an association matrix:

$$\begin{bmatrix} X & 2 & 5 & X \\ 6 & X & X & 3 \\ 8 & 3 & X & 4 \end{bmatrix} \qquad 5.1\text{-}2$$

In this example we have three tracks and four measurements. For simplicity we have used integers for the distances, but of course this will not generally be the case. The symbol "X" is used to indicate an unfeasible association.

The simplest method to find the optimal set of associations is a systematic search. For a relatively simple association matrix, this is not a bad method. In the example above, one can readily see by inspection that the optimal association assigns the 2nd measurement to the 3rd track (distance of 3), the 3rd measurement to the 1st track (distance of 5) and the 4th measurement to the 2nd track (distance of 3), for a total distance of 11, leaving the 1st measurement unassociated.

In general, the systematic search can be a lengthy process, because the number of possible assignment decisions (without reference to gate validation) is $a!/(a-b)!$ where a and b are respectively the greater and lesser of the number of rows and number of columns. Naïve methods exist, which are straightforward to implement but often do not lead to the optimal solution[11]. Fortunately, there are algorithms to find the optimal assignment that are much more efficient than the systematic search. One of these, the Munkres algorithm, is presented in Annex B. The time required for this algorithm is in the order of a^2b. The JVC (Jonker-Volgenant-Castanon) algorithm is reported to be even faster [7].

There are variations on NN methods in which assignment decisions pertaining to time k may be deferred until some later time. For example, one could use a sliding window of length c (some predetermined number of steps, a small positive integer) such that information from times $k+1$ through $k+c$ may be used to make the assignment decisions for time k. Such deferred-decision NN methods will not be dealt with in this document. Reference [1] gives a sampling of variations on the NN theme. For the ultimate deferred-decision algorithm refer to section 5.3.

---

[11] One example of a naïve method is to start with the smallest element of the association matrix and make the corresponding association, then continue by finding the smallest remaining element whose row and column are both still available, and so on until the possible valid associations are exhausted. A quick look at the example matrix 5.1-2 shows that this method will lead in this case to a total distance of 13, clearly not the optimal solution. Some writers reserve the term "nearest-neighbour" for this simplified method, while referring to the optimal NN association by the name of the specific algorithm for finding the optimal solution, such as JVC.

5.2 The Joint Probabilistic Data Association Filter

The JPDAF is the multi-target generalization of the PDAF. It assumes that we have a known, fixed number of targets for which tracks are already established, as well as possible false alarms, which are uniformly distributed throughout the surveillance region. In section 3.3 it was observed that the performance of the PDAF is degraded by persistent (non-random) interference, such as returns from a real object in the vicinity of the target. The JPDAF suffers similarly in the presence of persistent interference from unknown objects, but interference between multiple known objects causes very little trouble as long as those known objects are all being tracked.

In this section we will present a version of the JPDAF in which each target has a single KF-style (or EKF-style) process model. A multiple-mode "IMMJPDAF" could presumably be worked out by generalizing the IMMPDAF according to the principles of this section, but presentation of such a method will not be attempted here.

For each target we need process model parameters $\mathbf{F}^i(k)$ and $\mathbf{Q}^i(k)$ and a detection probability P_D^i. The target index (superscript) runs from 1 to N_T, the number of targets. We also need (as usual) the measurement model parameters $\mathbf{H}(k)$ and $\mathbf{R}(k)$.

This presentation of the algorithm is based on [1]. Starting the iteration with the state vectors $\hat{\mathbf{x}}^i(k-1|k-1)$ and their covariances $\mathbf{P}^i(k-1|k-1)$, we calculate the predictions $\hat{\mathbf{x}}^i(k|k-1)$ and $\mathbf{P}^i(k|k-1)$, along with the innovation covariances $\mathbf{S}^i(k)$ and gains $\mathbf{W}^i(k)$, as in an ordinary KF (or EKF). Let $M(k)$ denote the total number of measurements, and denote the measurements by $\mathbf{z}_j(k)$. The innovations are $\nu_j^i(k) = \mathbf{z}_j(k) - \mathbf{H}(k)\hat{\mathbf{x}}^i(k|k-1)$.

To proceed further in the iteration, we need to consider all the *association hypotheses*. An association hypothesis θ consists of assertions about which measurement belongs to which target (if any); it can be represented by its event matrix $\omega(\theta)$ whose components $\omega_j^i(\theta)$ are binary (each equal to 0 or 1). They have the following properties:

- i runs from 0 to N_T.
- j runs from 1 to $M(k)$.
- For $i > 0$, $\omega_j^i(\theta) = 1$ if θ asserts that measurement j belongs to target i.
- For $i > 0$, $\omega_j^i(\theta) = 0$ if θ asserts that measurement j does not belong to target i.
- $\omega_j^0(\theta) = 1$ if θ asserts that measurement j is a false alarm.
- $\omega_j^0(\theta) = 0$ if θ asserts that measurement j is not a false alarm.

The constraints on a hypothesis are:

-Each measurement has exactly one origin: either it is a false alarm, or it is from one target.

$$\sum_{i=0}^{N_T} \omega_j^i(\theta) = 1 \quad \forall j \qquad 5.2\text{-}1$$

-Each target is responsible for at most one measurement.

$$\sum_{j=1}^{M(k)} \omega_j^i(\theta) \le 1 \quad \forall i > 0 \qquad 5.2\text{-}2$$

-There can be any number of false alarms.

It is convenient to denote the second summation above as $\delta^i(\theta)$. This is called the target detection indicator, since it indicates whether target i has been detected or not. Let $t_j(\theta)$ denote the target index that is associated with measurement j in hypothesis θ, i.e., the number having the property that $\omega_j^{t_j(\theta)}(\theta) = 1$. Let $\phi(\theta)$ denote the number of false alarms according to θ:

$$\phi(\theta) = \sum_{j=1}^{M(k)} \omega_j^0(\theta). \qquad 5.2\text{-}3$$

Let $\Theta(k)$ denote the set of all feasible hypotheses at time k. To be considered feasible, a hypothesis must obey the constraints given above. We can further restrict $\Theta(k)$ to those hypotheses that obey every target's association gate: i.e., $\nu_j^{t_j(\theta)}(k)^T \mathbf{S}^{t_j(\theta)}(k)^{-1} \nu_j^{t_j(\theta)}(k) \le \gamma$ whenever $t_j(\theta) > 0$. Let $\Theta_j^i(k)$ denote the subset of $\Theta(k)$ consisting of hypotheses for which measurement j is associated with target i; i.e., $t_j(\theta) = i$. let $\Theta_0^i(k)$ denote the subset of $\Theta(k)$ for which no measurements are associated with target i.

Now we are ready to proceed to the next step of the JPDAF iteration. The association probabilities are

$$\beta_j^i = \frac{\sum_{\theta \in \Theta_j^i(k)} e(\theta)}{\sum_{\theta \in \Theta(k)} e(\theta)} \quad i = 1,\ldots,N_T \quad j = 0,\ldots,M(k) \qquad 5.2\text{-}4$$

(note the change in the range of j for this formula only). The summands are given by:

$$e(\theta) = \phi(\theta)! \prod_{j:t_j(\theta)>0} \left[\frac{V \exp\left(-\frac{1}{2} \nu_j^{t_j(\theta)}(k)^T \mathbf{S}^{t_j(\theta)}(k)^{-1} \nu_j^{t_j(\theta)}(k)\right)}{\sqrt{\det\left(2\pi \mathbf{S}^{t_j(\theta)}(k)\right)}} \right] \prod_{i=1}^{N_T} \left[\left(P_D^i\right)^{\delta^i(\theta)} \left(1 - P_D^i\right)^{1-\delta^i(\theta)} \right] \qquad 5.2\text{-}5$$

for the nonparametric JPDAF. Note that the second product is over all targets, while the first product is over all measurements that the hypothesis considers to be from a target (i.e., not from a false alarm). Here V is the volume of the entire surveillance region. (Note the similarity to the case in the PDAF when the gate volume is too complicated to calculate; the entire surveillance region volume and the total number of measurements are used.) If the

false alarm density λ is known, then the parametric JPDAF, with the following formula, is preferable:

$$e(\theta) = \prod_{t:\delta^t(\theta)>0} \left[\frac{\exp\left(-\frac{1}{2}\nu_j^{t,(\theta)}(k)^{\mathrm{T}}\mathbf{S}^{t,(\theta)}(k)^{-1}\nu_j^{t,(\theta)}(k)\right)}{\lambda\sqrt{\det\left(2\pi\mathbf{S}^{t,(\theta)}(k)\right)}} \right] \prod_{t=1}^{N_{\mathrm{T}}} \left[\left(P_{\mathrm{D}}^t\right)^{\delta^t(\theta)} \left(1-P_{\mathrm{D}}^t\right)^{1-\delta^t(\theta)} \right] \qquad 5.2\text{-}6$$

This change of formula is the multi-target analog for the substitution $\lambda = \frac{M(k)}{V(k)}$ that we saw in the single target case (section 3.3). Note that these two formulae for $e(\theta)$ are not equivalent; use one or the other, not both.

The iteration is finished off by calculating the combined innovation $\nu^t(k)$ and thence the updated state estimate $\hat{\mathbf{x}}^t(k \mid k)$ and its covariance $\mathbf{P}^t(k \mid k)$ as in the ordinary PDAF. (Use equations 3.3.1-3, with a target index attached to each variable.)

The JPDAF generally has better results than the NN methods when the latter are combined with the same KF or EKF that is used for the filtering aspect of the JPDAF.

5.3 Multiple-Hypothesis Tracking

The MHT method is the most complex and, in principle, the most effective of the tracking algorithms.

We encountered the notion of association hypotheses in the discussion of the JPDAF in section 5.2. There, the hypotheses consisted only of assertions related to the association of new measurements with established tracks or with false alarms. All past information was contained in the state vectors $\hat{\mathbf{x}}^t(k-1 \mid k-1)$ and their covariances $\mathbf{P}^t(k-1 \mid k-1)$, so the calculation of the association probabilities β_j^t used only this information. In the MHT, on the other hand, much more detailed information is held over from previous iterations. The hypotheses contain assertions about past associations as well as present associations. In principle, *all* available information is kept forever, and for this reason the MHT is often treated in the literature as the ultimate tracking method. However, each old hypothesis branches out into several new hypotheses at each iteration of the algorithm, so the number of hypotheses to be handled may grow exponentially with time. The limits of one's computational resources will inevitably be reached. Therefore, one is forced to adopt techniques to reduce the amount of data being held.

A "plain" MHT algorithm will be presented first (based on [1] and [2]) in some detail, with no consideration given to the data-trimming methods that are necessary in order to make the MHT practical. Afterwards, there will be a brief discussion of some of those data-trimming methods.

5.3.1 Set-up and Definitions

In the MHT method as presented here, all targets use the same process model, and that process model is of the type used in an ordinary KF (or EKF). Also, each target has the same

assumed probability of detection P_D. (The method is easily extended to cover a list of anticipated target types, each type having its own process model and probability of detection.) No attempt will be made here to present an MHT with multiple-mode process models.

It is useful to have an estimated (or assumed) false alarm density λ_F and new target density λ_N. The assumption can then be made that the number of false alarms, or the number of new targets, at each iteration will be, for example, Poisson-distributed. Without this information, we will have to assume an equal prior probability of every possible number of false alarms and of new targets, as in the nonparametric version of the PDAF or JPDAF.

After each iteration, the MHT retains some number of hypotheses. Each hypothesis θ has a time tag $\chi(\theta)$ indicating the time at which θ is up-to-date. That is, θ makes use of sensor data up to and including time $\chi(\theta)$; so for all hypotheses formed at time k, we have $\chi(\theta) = k$. Each hypothesis also has a probability $P(\theta)$, interpreted as the probability that θ is a true description of the system (in the sense that it made use of the correct association decisions) given the sensor data up to time $\chi(\theta)$.

Besides a time tag and a probability score, each hypothesis contains some number of tracks. Each hypothesis can have its own track-numbering system, independent of most other hypotheses; so, for example, track number 3 in one hypothesis need not in general have any relationship to track number 3 in another hypothesis. Exceptions (i.e., cases where the track numbering systems of different hypotheses are related) will be noted. The notation $i \in \theta$ is used to indicate that θ includes a track whose index is i.

Each track has a state estimate and a corresponding covariance matrix. We will use $\hat{\mathbf{x}}_\theta^i$ to denote the state estimate for the i^{th} track in θ, and $\mathbf{P}_\theta^i$ for its covariance. In order to make explicit the time-dependence of the state estimate and covariance (and thus keep our notation more consistent with what has gone before), we may write these as $\hat{\mathbf{x}}_\theta^i(\chi(\theta) \mid \chi(\theta))$ and $\mathbf{P}_\theta^i(\chi(\theta) \mid \chi(\theta))$.

5.3.2 The Iteration

Assume we have just finished the iteration for time $k-1$, and that we have a number $M(k)$ of measurements $\mathbf{z}_j(k)$ for time k. For each track in each hypothesis θ' from time $k-1$ (i.e., for which $\chi(\theta') = k-1$), we can calculate the predictions $\hat{\mathbf{x}}_{\theta'}^i(k \mid k-1)$ and $\mathbf{P}_{\theta'}^i(k \mid k-1)$, along with the innovation covariances $\mathbf{S}_{\theta'}^i(k)$ and gains $\mathbf{W}_{\theta'}^i(k)$, as in an ordinary KF (or EKF). The innovations are $\nu_j^i(k) = \mathbf{z}_j(k) - \mathbf{H}(k)\hat{\mathbf{x}}^i(k \mid k-1)$.

We must then construct all the feasible hypotheses for time k, by constructing all feasible successors to each hypothesis of time $k-1$. A hypothesis θ is a feasible successor to the hypothesis θ' if the following three conditions hold. First, $\chi(\theta) = \chi(\theta') + 1$. Second, each

measurement $\mathbf{z}_j(\chi(\theta))$ is deemed to be associated with exactly one existing track in θ', or to be a false alarm, or to represent a new target (these three possibilities being mutually exclusive). Third, each track has at most one new measurement associated with it. In order to keep the computational load down a little, it is reasonable to impose also the gate condition; i.e., the j^{th} measurement can be associated with the i^{th} track only if $\nu_j^i(\chi(\theta))^{\mathrm{T}}\mathbf{S}_{\theta'}^i(\chi(\theta))\nu_j^i(\chi(\theta)) \le \gamma$. Let $\zeta(\theta)$ denote the predecessor of θ, so that the equation $\zeta(\theta) = \theta'$ is equivalent to the statement that θ is a successor to θ'. In order to keep the algorithm (and its notation) manageable, it will be assumed that if $\zeta(\theta) = \theta'$ then the track numbering systems of θ and θ' agree as far as possible. Thus, if there are (for example) N tracks in θ', numbered from 1 to N, then the tracks numbered 1 to N in θ are updates of the same N tracks. There will be as many additional tracks in θ (presumably numbered from N+1 onward) as there are measurements that θ considers to belong to new targets.

For each new hypothesis θ let $\phi(\theta)$ denote the number of new measurements that are considered to be false alarms, and let $\nu(\theta)$ denote the number of new measurements that are considered to belong to new targets. (That it has an hypothesis as an argument, rather than a time, will remind us that this ν is not referring to an innovation.) Let $t_j(\theta)$ denote the index of the target with which the j^{th} measurement is associated according to θ. (Let $t_j(\theta) = 0$ if the j^{th} measurement was deemed a false alarm.) Let $\delta^i(\theta)$ be a target detection indicator; i.e., $\delta^i(\theta) = 1$ if the i^{th} track had an associated measurement according to θ, and $\delta^i(\theta) = 0$ otherwise.

Having thus constructed all the feasible hypotheses θ for which $\chi(\theta) = k$, the state estimates and covariances for these hypotheses must be calculated. If $t_j(\theta) = i$ and $i \in \zeta(\theta)$ (i.e., the i^{th} track in θ existed in $\zeta(\theta)$ and was associated with the measurement $\mathbf{z}_j(k)$), then $\hat{\mathbf{x}}_\theta^i(k \mid k)$ and $\mathbf{P}_\theta^i(k \mid k)$ are calculated as in an ordinary KF (or EKF), using $\nu_j^i(k)$ for the innovation. That is,

$$\begin{aligned}\hat{\mathbf{x}}_\theta^i(k \mid k) &= \hat{\mathbf{x}}_{\zeta(\theta)}^i(k \mid k-1) + \mathbf{W}_{\zeta(\theta)}^i(k)\nu_j^i(k),\\ \mathbf{P}_\theta^i(k \mid k) &= \mathbf{P}_{\zeta(\theta)}^i(k \mid k-1) - \mathbf{W}_{\zeta(\theta)}^i(k)\mathbf{S}_{\zeta(\theta)}^i(k)\mathbf{W}_{\zeta(\theta)}^i(k)^{\mathrm{T}}.\end{aligned} \qquad 5.3.2\text{-}1$$

If $i \in \zeta(\theta)$ but $\delta^i(\theta) = 0$ (i.e., the i^{th} track in θ existed in $\zeta(\theta)$ but was not associated with any measurement at time k), then $\hat{\mathbf{x}}_\theta^i(k \mid k) = \hat{\mathbf{x}}_{\zeta(\theta)}^i(k \mid k-1)$ and $\mathbf{P}_\theta^i(k \mid k) = \mathbf{P}_{\zeta(\theta)}^i(k \mid k-1)$. (In effect, there is no gain.) If $i \in \theta$ but $i \notin \zeta(\theta)$ (i.e., the i^{th} track in θ was considered to represent a new target), then the state estimate is initiated according to the procedure of section 4.3.

It remains to calculate the probabilities $P(\theta)$ for each new hypothesis.

$$P(\theta) = \frac{\varepsilon(\theta)}{\sum_{\theta} \varepsilon(\theta)} \qquad 5.3.2\text{-}2$$

where the sum in the denominator is over all feasible hypotheses. The $\varepsilon(\theta)$ terms are given by

$$\varepsilon(\theta) = P(\zeta(\theta))\phi(\theta)!\nu(\theta)! \times \prod_{j:\, j(\theta)\in\zeta(\theta)} \left[\frac{V \exp\left(-\frac{1}{2} \nu_j^{t_j(\theta)}(k)^{\mathrm{T}} \mathbf{S}_{\zeta(\theta)}^{t_j(\theta)}(k)^{-1} \nu_j^{t_j(\theta)}(k)\right)}{\sqrt{\det\left(2\pi \mathbf{S}_{\zeta(\theta)}^{t_j(\theta)}(k)\right)}} \right] \prod_{m\in\zeta(\theta)} \left[P_{\mathrm{D}}^{\delta_m(\theta)} (1-P_{\mathrm{D}})^{1-\delta_m(\theta)} \right] \qquad 5.3.2\text{-}3$$

if the false alarm density λ_{F} and new target density λ_{N} are unavailable. This case is analogous to the "nonparametric" (J)PDAF. (V is again the volume of the entire surveillance region.) The "parametric" case, where the false alarm density and the new target density are known (or guessed), is given by

$$\varepsilon(\theta) = P(\zeta(\theta))\lambda_{\mathrm{F}}^{\phi(\theta)} \lambda_{\mathrm{N}}^{\nu(\theta)} \times \prod_{j:\, j(\theta)\in\zeta(\theta)} \left[\frac{\exp\left(-\frac{1}{2} \nu_j^{t_j(\theta)}(k)^{\mathrm{T}} \mathbf{S}_{\zeta(\theta)}^{t_j(\theta)}(k)^{-1} \nu_j^{t_j(\theta)}(k)\right)}{\sqrt{\det\left(2\pi \mathbf{S}_{\zeta(\theta)}^{t_j(\theta)}(k)\right)}} \right] \prod_{m\in\zeta(\theta)} \left[P_{\mathrm{D}}^{\delta_m(\theta)} (1-P_{\mathrm{D}})^{1-\delta_m(\theta)} \right] \qquad 5.3.2\text{-}4$$

if a Poisson distribution is assumed for both the number of new targets and the number of false alarms. Note that these two formulae are not equivalent, so only one at a time should be used.

5.3.3 Hypothesis Management

Since the number of hypotheses tends to grow exponentially with time, any computer's resources will soon be exhausted, leading to a catastrophic failure of our tracking system. In order to make MHT practical, we need to impose a set of rules for reducing the number of hypotheses being retained. The more common schemes are described here (in outline only) [1] [2] [7].

Since each hypothesis carries with it a probability score $P(\theta)$, one can simply monitor the individual hypothesis probabilities and throw away any hypothesis whose probability has dropped below some pre-set threshold. In this case the probabilities of the remaining hypotheses will have to be renormalized so that their sum is unity. The only detail to be filled in for this procedure is the choice of threshold.

It is also possible that several hypotheses that are based on quite different past histories may nevertheless contain very similar results, including the same number of tracks, with very similar state estimates. In such a case, it is appropriate to merge the similar hypotheses into a single hypothesis whose probability is the sum of the probabilities of the hypotheses of which it was made. For such a procedure, one must make the notion of "very similar state estimates" more precise, and must work out how to appropriately average the state vectors into the merged state vector.

There are more sophisticated ways of deciding when to prune or merge. See, for example, [8] and references therein.

Cluster management is another important idea. Computational demands may be greatly eased by keeping tracks sorted into *clusters* consisting of all tracks that have shared measurements (i.e., measurements that belong to different tracks according to different hypotheses). Any track formed from a new measurement that is outside the gates for all existing tracks (in all hypotheses) forms a new cluster. Whenever a measurement is associated with tracks from more than one cluster, those clusters are merged. Whenever a set of measurements and their associations allows for the division of a cluster into two or more smaller clusters, this is done.

We may now consider separate hypotheses for separate clusters. The basic MHT as presented in subsection 5.3.2 considered global hypotheses, i.e., hypotheses that had something to say about every measurement. With the use of clusters, we may have one set of hypotheses (probabilities adding to unity) for one cluster, another (independent) set of hypotheses (probabilities adding to unity) for another cluster, and so on. By picking one hypothesis from each set, and multiplying the probabilities, we can reconstruct any one of the global hypotheses that we would have had if we had not used clustering. In this way the total number of hypotheses to be considered is greatly reduced. For example, if we have two clusters, the first one currently having a hypotheses and the second having b, our system must handle a total of $a+b$ hypotheses at this time, when the number of equivalent global hypotheses is $a \times b$.

The MHT procedure, as modified by the use of clusters, runs as follows: (1) Gather measurements. (2) Form new clusters and merge or split existing clusters as necessary, making corresponding modifications to existing hypotheses. (3) For each cluster, do the basic MHT iteration as in subsection 5.3.2, and then prune or merge hypotheses as appropriate.

5.4 Summary

In this chapter, we considered methods for tracking multiple targets in an environment in which false alarms can also arise.

The Nearest-Neighbour method (section 5.1) is a simple association algorithm that combines easily with any tracking algorithm and with any track-formation algorithm.

The Joint Probabilistic Data Association Filter (JPDAF, section 5.2) combines the roles of an association algorithm and a tracking algorithm. A Kalman-Filter-like JPDAF was presented. To put other tracking algorithms into the JPDAF framework is possible, but such a procedure was not addressed. The JPDAF is designed for a fixed number of targets. Complications may arise when trying to add a track-formation algorithm. Otherwise, it is superior to the Nearest-Neighbour method.

The Multiple-Hypothesis Tracking algorithm (section 5.3) is very sophisticated and potentially very effective but computationally intensive. Data trimming methods are required in order to render it practical. It combines the functions of association, tracking, and track formation.

6. Identity Data Fusion

Up to this point we have only considered kinematic data fusion, which addresses a target's position and change in position. In this chapter we consider identity data fusion, which addresses the question of what a target is. Information about a target's identity generally derives from attribute data. For example, identity information may arise from the set of acoustic frequencies detected by a passive sonar, from the set of electromagnetic frequencies picked up by ESM[12], by a target's IFF[13] response, or even by human interpretation of ordinary visual sensation.

Our data fusion system will need to be able to represent identity propositions of varying degrees of specificity. Ideally, the system will include a database (called a Platform Database, or PDB) of all conceivably possible types of targets. "The quality and performance of the identification process is much more a function of the completeness of the Platform Database than anything else." [7] An assertion about a target's identity corresponds to a subset of the target types in the PDB. Such an assertion can then be represented by a series of bits, one for each type of target in the PDB, where each bit that is turned "on" corresponds to a target type that is included in the subset, and each bit that is turned "off" corresponds to a target type that is excluded. It is thus easy to discern the logical relationship between two assertions about a given target – such as whether they are mutually contradictory, or whether one is implied by the other, and so on.

There are several possible approaches to representing uncertainty in identity data and combining such data from different sources. This chapter will deal only with the Dempster-Shafer method, which, according to its proponents (e.g., [9] or [2]), has several advantages over the classical (Bayesian) probabilistic approach, one of which is a more natural representation of ignorance. Section 6.1 describes the representation of its uncertainty in the Dempster-Shafer scheme, while section 6.2 presents the rule according to which independent sets of evidence are combined. The formalism of both sections is from [9]. Section 6.3 briefly discusses the combination of this information with kinematic information.

6.1 Basic Probability Assignments

An identity estimate in the Dempster-Shafer theory can be represented by a *belief function*[14] whose domain is the set of subsets of the full list of target types and whose range is the interval $[0,1]$, representing our degree of confidence that the target in question belongs to a

[12] ESM: Electronic Support Measures (passive detection of electromagnetic emissions).

[13] IFF: Identification of Friend or Foe; an automatic interrogation system built into many radar systems. The detection-and-response side of the IFF system is installed on many vehicles, so that (for example) commercial aircraft may identify themselves as such. The capacity to identify oneself as "friend" (a response one hopes to get from military units of one's own side, and not to get from those of the enemy) depends upon having access to the appropriate codes. Therefore a false "friend" response is assumed to be unlikely.

[14] The requirements for such a function to be properly considered a belief function (see [9] for details) are automatically fulfilled as a result of the relationship between belief functions and basic probability assignments described here.

given subset of target types. The belief functions of interest to us are those that correspond to (that is, can be derived from) a *basic probability assignment* (BPA), which is another function with the same domain and range as a belief function. A valid BPA, here denoted m, has the following constraints:

$$m(\varnothing) = 0,$$
$$\sum_{A \subseteq \Theta} m(A) = 1 \qquad 6.1\text{-}1$$

where Θ is the set of all target types. The belief function Bel is related to the BPA by:

$$Bel(A) = \sum_{B \subseteq A} m(B). \qquad 6.1\text{-}2$$

We can interpret $Bel(A)$ as a lower bound on the probability that the target in question belongs to the subset A of target types. $m(A)$ is the amount of probability that is directly assigned to the subset A. It contributes to our belief in A and to our belief in any superset of A. Notice that probability that is not assigned to a given proposition need not be assigned to its negation. Indeed, in practice a BPA always includes some nonzero value for $m(\Theta)$, representing our ignorance.

The interpretation of the belief function as a lower bound on the probability of a target belonging to a given set leads us to a definition of an *upper probability* function, to be interpreted as the corresponding upper bound on the probability:

$$P^*(A) = 1 - Bel(\overline{A}) \qquad 6.1\text{-}3$$

where $\overline{A}$ is the complement (negation) of A. Just as our belief in a given proposition (that the target belongs to a given set of types) represents the extent to which the BPA *implies* that proposition, the upper probability of a given proposition represents the extent to which the BPA *allows* that proposition. The belief and the upper probability are more commonly called the *support* and *plausibility*, respectively, and we will use this terminology from this point on.

In the simplest case, an identity estimate may take the form of a BPA which assigns a nonzero probability to only two sets, one of which is Θ. For example, the statement that "the target is a submarine" with a confidence of 0.90 would be represented by a BPA that assigns a value of 0.90 to the set of all submarines and 0.10 to the ignorance (i.e., to the full set of target types). In this case, the proposition that the target is a submarine would have a support of 0.90 and a plausibility of 1.00.

As a more complicated example of support and plausibility, suppose that our BPA assigns a value of 0.50 to submarines in general, 0.25 to hostile submarines, 0.15 to surface ships, and 0.10 to ignorance. Then the proposition that the target is a submarine has a support of 0.75 (since this proposition is implied by both the BPA value for submarines and the more specific BPA value for hostile submarines) and a plausibility of 0.85 (since this proposition is also allowed by our ignorance, which indeed allows everything). From equation 6.1-3 we can immediately see that the proposition that the target is *not* a submarine has support 0.15 and plausibility 0.25.

6.2 Dempster's Rule

Here we assume that we have two independent sets of evidence about the identity of a given target, and that each set of evidence is represented by a BPA (and hence by the associated support and plausibility functions) as in section 6.1. In this section we present the rule (known as Dempster's Rule) for combining the two sets of evidence into a single set of evidence, represented by another BPA.

Let the first BPA be denoted m_1 and the second m_2. Let

$$\kappa = \sum_{\substack{A,B\subseteq\Theta \\ A\cap B=\varnothing}} m_1(A)m_2(B) \qquad 6.2\text{-}1$$

represent the degree of logical conflict between the two BPAs. We will refer to κ simply as the *conflict*. (The practice of always admitting some degree of ignorance ensures that the conflict will be less than one.) Then the combined BPA is

$$m_{comb}(A) = \frac{1}{1-\kappa} \sum_{\substack{B,C\subseteq\Theta \\ B\cap C=A}} m_1(B)m_2(C) \qquad 6.2\text{-}2$$

for $A \neq \varnothing$, and $m_{comb}(\varnothing) = 0$. It is easy to see that m_{comb} obeys equation 6.1-1.

As an example, let m_1 be the BPA associated with the last example of section 6.1. Suppose we want to combine this BPA with another BPA, m_2, that assigns 0.60 to submarines, 0.10 to whales, and 0.30 to ignorance. The nonzero terms in the summation of equation 6.2-1 are the contradictions between m_1 and m_2, which are "submarine" vs "whale" (0.05), "hostile submarine" vs "whale" (0.025), "surface ship" vs "whale" (0.015), and "surface ship" vs "submarine" (0.09), for a total conflict of 0.18. The sets of target types for which the combined BPA will have a nonzero value are the intersections of noncontradictory sets from the two previous BPAs. These are "submarine", "hostile submarine", "surface ship", "whale", and "ignorance". The nonzero terms in the summation of 6.2-2 leading to the set "submarine" come from the combinations "submarine" with "submarine" (0.30), "submarine" with "ignorance" (0.15), and "ignorance" with "submarine" (0.06), leading to a final BPA for "submarine" of 0.51/(1 - 0.18) = 0.622. Similarly, the other nonzero final BPA values are 0.274 for "hostile submarine", 0.055 for "surface ship", 0.012 for "whale" and 0.037 for "ignorance". Thus the proposition that the target is a submarine now has a support of 0.896 and a plausibility of 0.933.

6.3 Fusing Identity Data with Kinematic Data

A complete presentation of a full kinematic-plus-identity fusion algorithm will not be attempted here. This section will merely outline the main issues involved in constructing such an algorithm. It is assumed that the identity-fusion aspect of our combined fusion algorithm will follow the Dempster-Shafer scheme.

Identity information impacts data association decisions. If identity information is available along with a kinematic measurement when calculating association probabilities in the JPDAF or MHT, a factor of $1-\kappa$ (where κ is the conflict between the identity information for the track and the identity information for the contact) should be inserted along with every instance

of $\exp(-\frac{1}{2}\nu\mathbf{S}^{-1}\nu)$. The corresponding modification to a NN algorithm is to add $-2\ln(1-\kappa)$ to the distance function [2].

Identity information impacts kinematic state updating, and vice-versa. Since different kinds of platforms move in different ways, a different process model could be assigned to each target type in our database. A variation on the IMM theme could be used, whereby the different manoeuvre modes could be identity-type-dependent. The identity information and the mode probabilities could be used to modify each other at each iteration.

7. Multiple Sensors

The appropriateness of the term *data fusion* becomes more apparent when we consider data coming from multiple sources. In this chapter we discuss some of the issues involved in combining data from several sensors, under the assumption that each sensor generates the kinds of data that we considered in previous chapters.

In the design of a data fusion system, there are fundamental choices to be made with respect to the flow of data, i.e., the system architecture. Section 7.1 will present some of the possible fusion architectures and their various strengths and weaknesses. Any choice of architecture other than a purely centralised one will necessitate track-level (i.e., track-to-track) fusion. Some of the issues involved in track-level fusion (essentially arising from error correlation) will be discussed in section 7.3. Section 7.2 deals with the potential problem of measurements being processed in a different order than that in which they were produced.

This is a convenient place to mention that, in order to combine data from multiple sensors, one (obvious) requirement is that the data have the same units and the same (positional and temporal) coordinate system. If the data are produced with different units or different coordinates, then the system must make the appropriate transformations. This process is called *alignment*. The term *registration* is sometimes used for positional alignment.

7.1 Data Fusion Architectures

In general, a data fusion system can be built up of subsystems that act as complete data fusion systems in their own right. Those subsystems will be referred to as *fusion nodes*. A fusion node is characterised by having contact and/or track data as input, maintaining a list of tracks by way of association and tracking algorithms, and having track data as output.

The algorithms discussed in previous chapters apply directly to a fusion node whose input consists of contact data from a single sensor. If a fusion node has access to contact data from multiple sensors, those algorithms need a few modifications, but these modifications are simple in principle. First, one must allow for different probabilities of detection and different measurement noises from the different sensors. This is not a problem for track maintenance, but the track formation and deletion decisions may have to be modified. (If sensor 1 persistently sees something that is invisible to sensor 2, but sensor 2 reports more frequently than sensor 1, an unmodified "2/2&m/n" algorithm may lead to no track being formed.) Second, contact data may be received out of sequence, as discussed in section 7.2. Third, association decisions may be more complicated if sensors are very dissimilar.

In practice, fusion nodes often take track data (i.e., the output of other nodes) as their input. This leads to other issues that are worth describing in some detail; see section 7.3.

There is a great variety of possible configurations of fusion nodes with respect to sensors [1] [2] [7] [10]. The configuration that is most straightforward to describe is the *centralised* architecture, in which there is only a single node to which all the contact data from all the sensors is made available. When there are multiple nodes, there will be connections that allow one node to send track data to another node. (The nodes and their connections may be

represented by a graph.) Such connections between nodes may be one-way or two-way. Here we will assume that one of the nodes will always be designated as that which provides the final output (and that the connections will allow the node so designated to be affected, whether directly or indirectly, by all the sensors in the system).

An architecture for which each sensor has a dedicated fusion node that uses contact data from that sensor (only) is called *sensor-based*. Some non-sensor-based multiple-node architectures may be referred to as *hybrid*, because they can be considered somewhere between centralised and sensor-based, but there is no strict boundary to this category.

If the graph representing the nodes and their connections is a tree (with the root node designated as providing the final output), then the architecture is *hierarchical*. Each node (except the root) will be able to send data to its immediate superior, one step toward the root. Hierarchical architectures may or may not allow feedback, i.e., data flow in the opposite direction (toward the leaves). A non-hierarchical architecture is usually referred to as *distributed* or *decentralised*. One possible distributed architecture has a dedicated fusion node for each sensor, and no other nodes at all; each node is connected to several other nodes (perhaps to all), and data may flow in both directions (or not). In this case, each node gets contact data only from its own sensor, but also gets track data from a variety of sources. In general a distributed architecture may have no obvious rhyme or reason to the way the nodes are connected.

Figures 1 to 4 depict a few examples of fusion architectures. In all cases the squares represent fusion nodes, the circles represent sensors, and the arrows represent the data flow. The arrow with the solid head represents the final output.

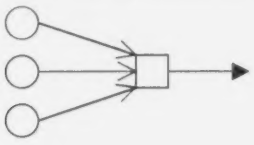


Figure 1. A centralised architecture.

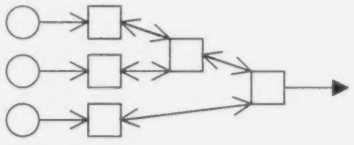


Figure 2. A sensor-based hierarchical architecture with feedback.

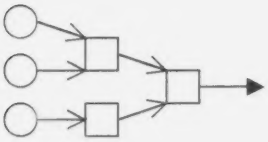


Figure 3. A hybrid hierarchical architecture without feedback.

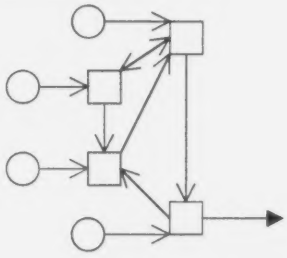


Figure 4. A sensor-based distributed architecture.

A track contains less information than all the contact data that were used to make that track. Therefore, centralised fusion is the best method in principle, since there is minimal

information loss. Centralised fusion will result in better association decisions and more accurate tracking, with tracks initialized and confirmed more quickly, and a lower incidence of lost tracks. There are, however, drawbacks to centralised fusion. There are high demands on both communications and computation. Also, the performance of the system may be strongly affected by sensor data that are degraded or corrupted (or misaligned), since tracking depends on the expected behaviour of the sensors.

The strengths and weaknesses of sensor-based fusion are of course the reverse of the weaknesses and strengths of centralised fusion. Tracks are formed on a sensor-by-sensor basis, so having some sensors corrupted will have less impact on tracking in a sensor-based architecture, and will be easier to notice. Sensor-based fusion also has the advantage that it naturally involves parallel processing. The load on each processor is less than in centralised fusion. On the other hand, the ultimate association and tracking results are suboptimal in general. There is also added complexity inherent in track-level fusion, as discussed in section 7.3.

In terms of robustness and flexibility, distributed data fusion is the most attractive option, because the addition or deletion of a node in midstream does not essentially change the nature of the architecture. Other sensor-based architectures also retain some ability to add or delete (some) nodes. Centralised fusion does not allow the addition of a node, and deletion of a node means deletion of the entire system. These characteristics are relevant in a tactical setting.

For a more detailed discussion of the strengths and weaknesses of different architectures, see [2] or [7].

7.2 Out-of-Sequence Measurements

It may happen that measurements may arrive (at the fusion node) out of sequence. In this case a measurement may arrive with a time-stamp that is earlier than the time up to which the tracks have been updated. In this case our tracking algorithm must be modified. A simple modification to the basic KF, in order to handle this situation, follows (taken from [1]). (Modifications to more complicated tracking algorithms are directly analogous.) Here we assume that the association decisions are unaffected.

Suppose that we have a track that has been updated up to time k (presumably using measurements up to that time), but that our next bundle of contact data to be associated with this track came from an earlier time. The time argument for the new measurement will still be represented as $k+1$, in honour of the processing sequence, but in this case time $k+1$ is earlier than time k. Ultimately, we want to use the new data to update the state of the track at time k.

A "predicted" state estimate and covariance for time $k+1$ is calculated in the usual way except that the process noise is dropped, since it has already been applied to all times up to k.

$$\begin{aligned} \hat{\mathbf{x}}(k+1 \mid k) &= \mathbf{F}(k)\hat{\mathbf{x}}(k \mid k) + \mathbf{G}(k)\mathbf{u}(k) \\ \mathbf{P}(k+1 \mid k) &= \mathbf{F}(k)\mathbf{P}(k \mid k)\mathbf{F}^{\mathrm{T}}(k) \end{aligned} \qquad 7.2\text{-}1$$

Note that the transition matrix $\mathbf{F}(k)$ depends implicitly on the length of the time interval, which in this case is negative. The innovation $\nu(k+1)$, the innovation covariance $\mathbf{S}(k+1)$, and the gain $\mathbf{W}(k+1)$ are calculated in the usual way. However, they are then applied to the state estimation at time k rather than at time $k+1$:

$$\begin{aligned}\hat{\mathbf{x}}(k \mid k+1) &= \hat{\mathbf{x}}(k \mid k) + \mathbf{W}(k+1)\nu(k+1) \\ \mathbf{P}(k \mid k+1) &= \mathbf{P}(k \mid k) - \mathbf{W}(k+1)\mathbf{S}(k+1)\mathbf{W}^{\mathrm{T}}(k+1)\end{aligned} \qquad 7.2\text{-}2$$

Further updates after $k+1$ will be applied with time k, rather than $k+1$, acting as the "former" time in the formulae.

7.3 Track-Level Fusion

This section is concerned with a description of some algorithms that could be used in a track-level fusion node, i.e., a node that takes track data as input. We will need to distinguish between the input tracks and the tracks that are created and maintained by the node under consideration. We will refer to the latter tracks as "internal tracks".

Subsection 7.3.1 briefly treats the problem of association as it applies to track-level fusion. Any unassociated input track can be used directly to create a new internal track. (If the track-level node uses the same track data structure that is used by the node submitting the input track, this track creation is a trivial matter of copying data.) Subsections 7.3.2 and 7.3.3 address the updating of an internal track in response to an input track after making an association. Track deletion need not be any different in principle for a track-level node than for a contact-level node, so no more will be said here about it.

In subsections 7.3.1 and 7.3.2 we will distinguish between the case where the input track and the internal track have previously been associated (and one has been used to create or update the other) and the case where the two tracks have not been so associated (such as when one of the tracks is newly-formed).

It has been mentioned that the fusion of tracks with other tracks involves complications that do not apply to the fusion of tracks with contacts. Most of these added complications are related to the fact that the errors in the tracks to be combined are usually not independent. It will be seen that optimal track-level track maintenance (subsection 7.3.2) is highly architecture-dependent and can be dreadfully complex. Subsection 7.3.3 presents the Covariance Intersection (CI) method as a conservative (and architecture-independent) method for updating a track in response to an input track.

In all cases we will assume that the input tracks and the internal tracks being updated are single-mode tracks, whose current state is described simply by a state estimate and covariance matrix.

7.3.1 Track-Level Association

This subsection is concerned with the case in which (track-level) node B receives a list of track data from node A, and must decide which of the tracks from A (if any) are to be associated with the internal tracks of B (if any). More specifically, we will be considering the pairwise comparison of an input track from node A, which has been updated to the current

time k, with an internal track of node B, which has previously been updated to an equal or earlier time k', then predicted (using the process model of B) to time k for the sake of comparison with the input tracks. We will express the state estimate and covariance of the input track as $\hat{\mathbf{x}}^{A}(k \mid k)$ and $\mathbf{P}^{A}(k \mid k)$, and those of the internal track as $\hat{\mathbf{x}}^{B}(k \mid k')$ and $\mathbf{P}^{B}(k \mid k')$.

As with contact-level nodes, a track-level node could have a wide variety of association schemes. Here we will only consider those of the immediate-decision type.

Case one: new input track

For our first case, suppose that the input track from node A and the internal track from node B have no data in common, i.e., neither has been used previously to update (or create) the other.

A typical approach [1] to this association problem is to use a statistical test, where the null hypothesis H_0 is the hypothesis that the pair of tracks (from two different lower-level nodes) do indeed represent the same target. The decision to accept or reject the hypothesis is based upon a comparison of the "distance" between the tracks with a threshold; that is, we reject H_0 if

$$\left(\hat{\mathbf{x}}^{A}(k \mid k) - \hat{\mathbf{x}}^{B}(k \mid k')\right)^{T} \left(\mathbf{T}^{AB}(k \mid k,k')\right)^{-1} \left(\hat{\mathbf{x}}^{A}(k \mid k) - \hat{\mathbf{x}}^{B}(k \mid k')\right) > D_{\alpha} \qquad 7.3.1\text{-}1$$

where $\mathbf{T}^{AB}(k \mid k,k')$ is the covariance[15] of $\hat{\mathbf{x}}^{A}(k \mid k) - \hat{\mathbf{x}}^{B}(k \mid k')$ (discussed below), and D_{α} is the threshold, which is set so that the probability, *assuming that* H_0 *is true*, of our rejecting H_0 (based on inequality 7.3.1-1) is equal to α (which can be set to, say, 0.01). Assuming further that $\hat{\mathbf{x}}^{A}(k \mid k) - \hat{\mathbf{x}}^{B}(k \mid k')$ is Gaussian (and that our covariance is correct), D_{α} is based on a chi-squared distribution with as many degrees of freedom as the number of dimensions in our state estimate, and can be looked up in statistical tables. For example, with a three-dimensional state estimate and $\alpha = 0.01$, D_{α} is about 11.34. Formally,

$$D_{\alpha} = \chi^{2}_{n}(1-\alpha). \qquad 7.3.1\text{-}2$$

If H_0 is not rejected (according to inequality 7.3.1-1) then the pair of tracks is a candidate for association. If a conflict arises (for example, because more than one track from node A can associate with a given track from node B) then a nearest-neighbour procedure to resolve the conflict is appropriate.

The covariance $\mathbf{T}^{AB}(k \mid k,k')$ can be expanded as

$$\mathbf{T}^{AB}(k \mid k,k') = \mathbf{P}^{A}(k \mid k) + \mathbf{P}^{B}(k \mid k') - \mathbf{P}^{AB}(k \mid k,k') - \mathbf{P}^{BA}(k \mid k',k) \qquad 7.3.1\text{-}3$$

where

---

[15] The notation used here for the time-dependence of $\mathbf{T}^{AB}(k \mid k,k')$ and $\mathbf{P}^{AB}(k \mid k,k')$ is non-standard but should be clear.

$$\mathbf{P}^{AB}(k \mid k,k') = \left\langle \left(\mathbf{x}(k) - \hat{\mathbf{x}}^{A}(k \mid k)\right)\left(\mathbf{x}(k) - \hat{\mathbf{x}}^{B}(k \mid k')\right)^{T} \right\rangle \qquad 7.3.1\text{-}4$$

is the cross-covariance between the state estimates $\hat{\mathbf{x}}^{A}(k \mid k)$ and $\hat{\mathbf{x}}^{B}(k \mid k')$. This cross-covariance is in general difficult to calculate. It arises despite the input track and internal track having no data in common, because both tracks have been affected by the same process noise. If both A and B have only just formed their tracks, then the cross-covariance may be negligible, but after a few updates it becomes significant. See [1] for a fairly thorough discussion of this matter. For purposes of association, the effect of the cross-covariance is to make the association test more stringent than it would have been if we had made the naïve assumption

$$\mathbf{T}^{AB}(k \mid k,k') = \mathbf{P}^{A}(k \mid k) + \mathbf{P}^{B}(k \mid k'). \qquad 7.3.1\text{-}5$$

Therefore, a practical approach is to adopt equation 7.3.1-5 in order to make $\mathbf{T}^{AB}(k \mid k,k')$ calculable, but to make up for leaving out the cross-covariance by increasing α to 0.02 or 0.03.

It has been assumed that the two lower-level tracks are commensurate, in the sense that their state estimates have the same number of dimensions and that those dimensions have the same interpretation. If this is not the case, then the two state estimates should be projected onto a common space and their covariances similarly restricted.

If the tracks from nodes A and B contain identity information, then any degree of contradiction between the two tracks (as in section 6.2) should be used to make the association test (7.3.1-1) more strict. For example, one might replace 7.3.1-2 with

$$D_{\alpha} = \chi^{2}_{n}\left((1-\alpha)(1-\kappa)\right) \qquad 7.3.1\text{-}6$$

Case two: previously-used input track

Now let us consider the case where one of the two tracks under consideration – either the input track from node A or the internal track from node B – has already been used to update (or create) the other track, most recently at time k'' (earlier than or equal to k'). Suppose also that the input track has been further updated to time k, and we are considering whether to use the input track (again) to make modifications to our internal track.

If the input track under consideration is the only track that has previously contributed to the internal track under consideration, then there is no question. We might even consider the internal track to be associated with the input track by definition.

If other input tracks, whether from node A or from other contributing nodes, have also been used in the creation or maintenance of the internal track under consideration, then the problem of association becomes more interesting. If we assume that the two tracks under consideration must be associated with each other as long as they both shall live, then there is the danger that we may not be able to recover from a (previous or eventual) association error. We could follow the procedure of case one, but the cross-covariance would then be so large that the simplification represented by equation 7.3.1-5 would no longer make sense in principle, and it would take some experimentation to find an appropriate value of α. If α is too small, it may take some time to detect any separation of the various input tracks, but if it is

too large, this may result in node B throwing away an input track as a result of a target manoeuvre – even after the lower-level node has processed that manoevre well.

Other procedures are possible, with additional bookkeeping. For example, if each internal track in node B keeps a record of which tracks from which other nodes (including A) have been used in creation and maintenance of the internal track, along with the last state estimate and covariance sent by each of those input tracks, then we can compare a newly-updated input track with each of the other previously-used tracks (taking the old, stored, state estimate and covariance and predicting to time k) via the procedure of case one. A track that differs too much from the predictions of the other contributing input tracks would then be rejected for association with the internal track under consideration. This procedure is appropriate for a hierarchical architecture without feedback, since the contributing (lower-level) nodes are not using any common data. Note that the data storage requirements are much greater for a multi-node architecture (if this procedure is used) than for a purely centralised (single-node) architecture. This extra data storage space may be necessary anyway, depending on the method used for track updating. See the following two subsections.

7.3.2 Track Maintenance in a Track-Level Node

In this subsection we will assume that all necessary association decisions have already been made, or else pose no problem. We will divide this subsection into two cases, corresponding to the two cases of subsection 7.3.1.

Case one: new input track

Once the decision is made to associate a new track from node A with an internal track of node B, as in case one of subsection 7.3.1, the updated state estimate and covariance of the internal track are naïvely (assuming no cross-covariance) given by

$$\begin{aligned}\hat{\mathbf{x}}^{\mathrm{B}}(k\mid k) &= \hat{\mathbf{x}}^{\mathrm{A}}(k\mid k)+\mathbf{P}^{\mathrm{A}}(k\mid k)\left(\mathbf{P}^{\mathrm{A}}(k\mid k)+\mathbf{P}^{\mathrm{B}}(k\mid k')\right)^{-1}\left(\hat{\mathbf{x}}^{\mathrm{B}}(k\mid k')-\hat{\mathbf{x}}^{\mathrm{A}}(k\mid k)\right)\\ \mathbf{P}^{\mathrm{B}}(k\mid k) &= \mathbf{P}^{\mathrm{A}}(k\mid k)-\mathbf{P}^{\mathrm{A}}(k\mid k)\left(\mathbf{P}^{\mathrm{A}}(k\mid k)+\mathbf{P}^{\mathrm{B}}(k\mid k')\right)^{-1}\mathbf{P}^{\mathrm{A}}(k\mid k)\end{aligned} \qquad 7.3.2\text{-}1$$

or, equivalently,

$$\begin{aligned}\mathbf{P}^{\mathrm{B}}(k\mid k)^{-1}\hat{\mathbf{x}}^{\mathrm{B}}(k\mid k) &= \mathbf{P}^{\mathrm{A}}(k\mid k)^{-1}\hat{\mathbf{x}}^{\mathrm{A}}(k\mid k)+\mathbf{P}^{\mathrm{B}}(k\mid k')^{-1}\hat{\mathbf{x}}^{\mathrm{B}}(k\mid k')\\ \mathbf{P}^{\mathrm{B}}(k\mid k)^{-1} &= \mathbf{P}^{\mathrm{A}}(k\mid k)^{-1}+\mathbf{P}^{\mathrm{B}}(k\mid k')^{-1}.\end{aligned} \qquad 7.3.2\text{-}2$$

With the cross-covariance included [1], the updated state estimate and covariance of the internal track are given by

$$\begin{aligned}\hat{\mathbf{x}}^{\mathrm{B}}(k\mid k) &= \hat{\mathbf{x}}^{\mathrm{A}}(k\mid k)+\left(\mathbf{P}^{\mathrm{A}}(k\mid k)-\mathbf{P}^{\mathrm{AB}}(k\mid k,k')\right)\\ &\quad\times\left(\mathbf{P}^{\mathrm{A}}(k\mid k)+\mathbf{P}^{\mathrm{B}}(k\mid k')-\mathbf{P}^{\mathrm{AB}}(k\mid k,k')-\mathbf{P}^{\mathrm{BA}}(k\mid k',k)\right)^{-1}\\ &\quad\times\left(\hat{\mathbf{x}}^{\mathrm{B}}(k\mid k')-\hat{\mathbf{x}}^{\mathrm{A}}(k\mid k)\right)\\ \mathbf{P}^{\mathrm{B}}(k\mid k) &= \mathbf{P}^{\mathrm{A}}(k\mid k)-\left(\mathbf{P}^{\mathrm{A}}(k\mid k)-\mathbf{P}^{\mathrm{AB}}(k\mid k,k')\right)\\ &\quad\times\left(\mathbf{P}^{\mathrm{A}}(k\mid k)+\mathbf{P}^{\mathrm{B}}(k\mid k')-\mathbf{P}^{\mathrm{AB}}(k\mid k,k')-\mathbf{P}^{\mathrm{BA}}(k\mid k',k)\right)^{-1}\\ &\quad\times\left(\mathbf{P}^{\mathrm{A}}(k\mid k)-\mathbf{P}^{\mathrm{AB}}(k\mid k,k')\right)\end{aligned} \qquad 7.3.2\text{-}3$$

The effect of the cross-covariance is to increase the fused covariance $\mathbf{P}^{B}(k \mid k)$, and to shift the fused estimate $\hat{\mathbf{x}}^{B}(k \mid k)$ slightly toward the more confident of $\hat{\mathbf{x}}^{A}(k \mid k)$ and $\hat{\mathbf{x}}^{B}(k \mid k')$. As noted in subsection 7.3.1, the cross-covariance is generally difficult to calculate. If new input tracks do not arrive frequently, one can get away with calculating the updated state estimate and covariance naïvely (by equations 7.3.2-1 or 7.3.2-2), then increasing the covariance, say by a factor of 1.2. But a better choice is to use the Covariance Intersection algorithm of subsection 7.3.3.

Case two: previously-used input track

The following discussion is based mostly on [10]. Following case two of subsection 7.3.1, we now suppose that we are using a track (newly updated to time k) from node A to update an internal track from node B (last updated to time k', earlier than k), and that the input track from A has previously been used (most recently at time k'', earlier than or equal to k') to update or create the internal track from node B. (The sub-case in which the previous sharing of data between these two tracks was in the other direction, from B to A, will be considered later.)

The optimally updated state estimate and covariance are given by

$$\begin{aligned} \mathbf{P}^{B}(k \mid k)^{-1} &= \mathbf{P}^{B}(k \mid k')^{-1} + \mathbf{P}^{A}(k \mid k)^{-1} - \mathbf{P}^{A}(k \mid k'')^{-1} \\ \mathbf{P}^{B}(k \mid k)^{-1}\hat{\mathbf{x}}^{B}(k \mid k) &= \mathbf{P}^{B}(k \mid k')^{-1}\hat{\mathbf{x}}^{B}(k \mid k') + \mathbf{P}^{A}(k \mid k)^{-1}\hat{\mathbf{x}}^{A}(k \mid k) \\ &\quad - \mathbf{P}^{A}(k \mid k'')^{-1}\hat{\mathbf{x}}^{A}(k \mid k'') \end{aligned} \qquad 7.3.2\text{-}4$$

where the predicted values, with time-dependences of $(k \mid k')$ or $(k \mid k'')$, are predicted using the $\mathbf{F}$ and $\mathbf{Q}$ matrices of the process models used by the appropriate nodes.

Without the final subtraction, equations 7.3.2-4 would be the same as 7.3.2-2. One can think of the inverse covariance matrices as representing amounts of "information", as in the information filter formulation of the KF (see equations 2.2-11). Node B is using the information it already has (the first term), along with the new information from node A (the second term). The third term represents the information that is shared by the first two terms; the subtraction is necessary in order to avoid double-counting this information. Without making this subtraction, one is essentially assuming that $\hat{\mathbf{x}}^{B}(k \mid k')$ and $\hat{\mathbf{x}}^{A}(k \mid k)$ are independent. This assumption can lead to rumour propagation and other disastrous results.

The procedure of equations 7.3.2-4 is sufficient to handle all the track-level fusion in a hierarchical sensor-based fusion architecture without feedback. Node B is the "superior" node, while any of the lower-level nodes can be represented by node A in 7.3.2-4.

Note that this procedure requires that past information be stored along with present information. For example, the track data structure in node B could include the state estimate and covariance (with the time-stamp) of the last data passed from each lower-level node. The data storage requirements are therefore much greater for a multi-node architecture (if this method is used) than for a purely centralised (single-node) architecture.

Feedback can be introduced into a hierarchical system in a straightforward way. Again, suppose that node B is using data from node A to update a track. Node B now refers to whichever node is getting data from the other at time k, regardless of whether it is the higher-level or lower-level node. If information can go in both directions between nodes A and B, then both A and B must store the state estimate and covariance (with time-stamp) that were last passed between them, regardless of direction. Equations 7.3.2-4 still apply, but the subtracted term (in both equations) is now taken to be the predicted (to time k) version of the last data exchanged (at time k^*) between A and B, regardless of direction. In other words, 7.3.2-4 applies unmodified if the last exchange between A and B was in the direction from A to B (i.e., in the same direction as the new data exchange at time k), but if the last exchange between these nodes was in the direction from B to A, the final term changes to $\mathbf{P}^B(k \mid k^*)^{-1}$ (for the first equation) and $\mathbf{P}^B(k \mid k^*)^{-1}\hat{\mathbf{x}}^B(k \mid k^*)$ (for the second equation).

So far in this subsection we have presented the optimal track-level fusion method in the case of strictly hierarchical architectures, with or without feedback. A distributed architecture is harder to handle, as equations 7.3.2-4 are not always correct in this case. In general there may be multiple data exchanges in the past, which need to be subtracted out of a new track update. Moreover, those pieces of information that are being subtracted out may themselves depend on common information, which would then have to be added back in. The essential idea behind 7.3.2-4 is still correct, but detailed information about past data exchanges must be available. It may be necessary to allow the nodes to query each other about their past history of data exchange. A general algorithm for getting the right subtractions and re-additions of shared information is complicated and its presentation will not be attempted here.

The methods discussed in this subsection allow a node to make the best possible use of new data from another node. Keeping track of data exchanged in the past is crucial, however. If the past data are not available, then other methods are necessary. If the cross-covariance is known (which is unlikely if the past exchanged information is not known), then equations 7.3.2-3 are nearly as good as 7.3.2-4 and its generalizations. It is not acceptable in general to use 7.3.2-1 or 7.3.2-2. When nothing is known about the shared information or cross-covariance, the method of subsection 7.3.3 is appropriate.

7.3.3 Covariance Intersection

The Covariance Intersection (CI) method, presented here, is appropriate when track data are being fused without knowledge of the degree of interdependence. As in subsection 7.3.2, we will assume that we are using a track (newly updated to time k) from node A to update an internal track from node B (last updated to time k', earlier than k). But now we do not care about the history of data exchanges between A and B. There is no need to divide into the two cases of subsections 7.3.1 and 7.3.2, since the method is the same in either case.

The CI equation for the covariance is (from [11])

$$\mathbf{P}^B(k \mid k)^{-1} = \omega\mathbf{P}^A(k \mid k)^{-1} + (1-\omega)\mathbf{P}^B(k \mid k')^{-1} \qquad 7.3.3\text{-}1$$

where the parameter ω is whatever real number in the interval $[0,1]$ minimizes the determinant[16] of $\mathbf{P}^{B}(k \mid k)$. (This parameter must be recalculated each time equation 7.3.3-1 is used. If one picks a value for ω and sticks with it, the algorithm may diverge.) Having found the appropriate value of ω, the state estimate is calculated by

$$\mathbf{P}^{B}(k \mid k)^{-1}\hat{\mathbf{x}}^{B}(k \mid k) = \omega\mathbf{P}^{A}(k \mid k)^{-1}\hat{\mathbf{x}}^{A}(k \mid k) + (1-\omega)\mathbf{P}^{B}(k \mid k')^{-1}\hat{\mathbf{x}}^{B}(k \mid k'). \qquad 7.3.3\text{-}2$$

It is worth discussing briefly and roughly what the CI method does. The degree of confidence of the updated state, as expressed by the covariance matrix, will always be similar to that of the more confident of the two input states, never higher. If one of the input states is much more confident than the other, the fused state estimate and covariance will be exactly equal to the more confident of the inputs. Otherwise, if the two input covariance matrices are similar to each other in the sense that they correspond to uncertainty ellipses that are of about the same shape and in about the same orientation, then the fused covariance will preserve this orientation, but if the two input covariances are not similar in this sense, then the fused covariance will have a shape and orientation representing a compromise between the two, and leaning toward the more confident.

The CI method can be considered cautious, even pessimistic. Its strengths are that it is unnecessary to know the degree of correlation between $\hat{\mathbf{x}}^{A}$ and $\hat{\mathbf{x}}^{B}$, but the fact that its output is never more confident than the more confident of the inputs ensures that no degree of correlation will cause it to fail. Its weakness is that its results may be underconfident. The standard track-level fusion method (of subsection 7.3.2) is preferable if the required information is available, but CI may be easier to apply in some cases (e.g., for distributed architectures). And whenever there is any uncertainty about correlations between data, CI is safer.

---

[16] It is possible to use other norms for $\mathbf{P}^{C}$ in order to calculate ω. For example, one could use the trace, or simply the (1,1)-element. But the determinant is the most natural choice.

8. Other Issues in Level One Data Fusion

In this chapter we present a rough list (with no claims regarding completeness) of some further matters of relevance to level one data fusion that have not been dealt with in previous chapters[17].

We have treated "detection" as a given throughout this document. In fact, the appropriate setting of a sensor's thresholds (to affect both the probability of detection and the probability of false alarm) is an important question. One's choice of such a threshold is measured and justified precisely by the eventual performance of the data fusion algorithms.

When using only passive sensors – for example, if an active sensor is unavailable, or tactical considerations preclude its use – then determining the range to a target is not straightforward. One possibility is to have multiple passive sensors acting together so that the range may be determined by triangulation. However, if there are multiple targets, and the targets and sensors are all in the same plane (or nearly so), then the presence of "ghosts" as shown in figure 5 may cause confusion. The black rectangles on the left represent the sensors, the squares are the targets, and the circles are the "ghosts" – possible erroneous positions for the targets.

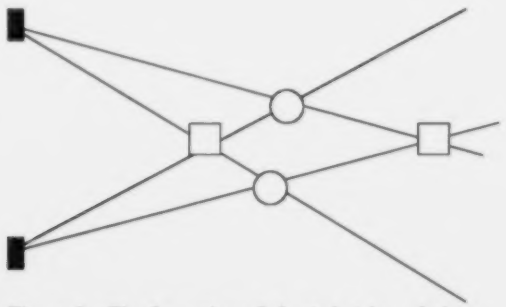


Figure 5 – The formation of ghosts in triangulation.

Alternatively, a platform with a single passive sensor may be able to gain range information on its own, if it manoeuvres while its target does not. This is common practice in Target Motion Analysis (TMA) using towed sonar arrays.

The use of sonar in maritime applications is further complicated by the fact that sound does not move in a straight line underwater. Moreover, there may be many transmission paths of sound from target to sensor.

Active sensors are vulnerable to a variety of countermeasures such as artificial returns designed to deceive the sensor by imitating echoes.

---

[17] The fact that these issues have been relegated to this chapter should not be construed as implying that they are unimportant, or that little work has been done on them.

Lastly, the kind of information to be gained by level one data fusion can, in principle, be refined by using a variety of types of non-real-time data. Knowledge of standard air corridors or common navigational routes, intelligence reports, announced travel plans, or military organisation and doctrine may all have the potential to enhance the results.

9. Conclusion

This document brings together into one place, for convenience, a collection of algorithms of level one data fusion, along with descriptions of when they are applicable. The selection of algorithms began with techniques for tracking the motion of a single target using input from a single sensor, under the assumption that there are no false alarms. The association question, as it relates to distinguishing a known real target from false alarms, was considered next. This was followed by a discussion of the formation and deletion of tracks, then the extension of the single-target tracking algorithms to the multiple-target case. Identity data fusion was considered briefly. Lastly, the case of multiple sensors was considered, along with a discussion of fusion architectures and the requirements of track-level fusion.

It is believed that the presentation of some of the algorithms (especially the IMMPDAF and the MHT) is significantly clearer here than in the references. As well, the content of this document goes beyond what can be gathered from the references in several cases, some of which are worth mentioning here. The polar-rectangular de-biasing formulae, available in [1] and [5] for two dimensions, were extended to three dimensions in section 2.4. Section 3.1 presented strategies for the construction of validation gates when multiple-mode filters are used. Equation 2.2-5 represents a correction to the process noise covariance in reference [5] (see footnote 4). Appendix B addresses a deficiency of the presentation of the Munkres algorithm in [2].

10. References

1. Y. Bar-Shalom, and X.-R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*, Yaakov Bar-Shalom, 1995.

2. S. S. Blackman, *Multiple-Target Tracking with Radar Applications*, Artech House, 1986.

3. D. L. Hall, and J. Llinas, "An Introduction to Multisensor Data Fusion", *Proc. IEEE*, **85**, pp. 6-23, 1997.

4. A. N. Steinberg, C. L. Bowman, and F. E. White, "Revisions to the JDL Data Fusion Model", *SPIE Proceedings*, **3719**, pp. 430-441, 1999.

5. J. M. J. Roy, "Demonstration of Kalman Filtering Techniques for Kinematics Data Fusion", DREV TR 1999-214, September 2000, UNCLASSIFIED

6. Y. Bar-Shalom, and T. E. Fortmann, *Tracking and Data Association*, Academic Press, 1988.

7. J. M. J. Roy, and É. Bossé, "A Generic Multi-Source Data Fusion System", DREV-R-9719, June 1998, UNCLASSIFIED

8. J. M. J. Roy, N. Duclos-Hindié, and D. Dessureault, "A Depth Control Pruning Mechanism for Multiple Hypothesis Tracking", *Fusion '99 International Conference*, paper C-085, 1999.

9. G. Shafer, *A Mathematical Theory of Evidence*, Princeton University Press, 1976.

10. C.-Y. Chong, "Distributed Architectures for Data Fusion", *Fusion '98 International Conference*, pp. 84-91, 1998.

11. J. K. Uhlmann, http://ait.nrl.navy.mil/people/uhlmann/CovInt.html

Annexes

A. Bias in Polar-Rectangular Coordinate Conversion

This annex consists of a partial derivation (including, it must be admitted, one seriously non-rigorous step) of the biases that arise in the standard polar-rectangular conversion (2.2-4).

Let us use r,α,θ to denote the spherical polar coordinates of the true position, and r_m,α_m,θ_m to denote the coordinates of the measured position. Let $\tilde{r},\tilde{\alpha},\tilde{\theta}$ denote the errors in these coordinates, so that $r_m = r + \tilde{r}$ etc. Let similar notation be used for the cartesian coordinates (but "measured" values in this case mean values derived from the measured polar coordinates via equation 2.4-2). We will concentrate on the x-coordinate. The other cartesian coordinates are handled similarly.

The bias in x is the mean error in x. In general, the bias will vary with position. We can calculate $\langle\tilde{x}\rangle|_{r,\alpha,\beta}$, the mean error conditioned on the true position, since knowledge of our sensors can give us knowledge of the probability distributions of the errors in the measured polar coordinates conditioned on the true position. Unfortunately, we want to know $\langle\tilde{x}\rangle|_{r_m,\alpha_m,\beta_m}$, the mean error conditioned on the measured position, and there is in general no way to know anything about the probability distributions of the errors in the measured polar coordinates conditioned on the measured position.

Assuming that the errors in the measured polar coordinates are zero-mean, Gaussian, and mutually uncorrelated (which is a fair assumption if we are considering them to be conditioned on the true position), we have

$$\langle\tilde{r}\rangle = 0, \quad \langle\sin\tilde{\alpha}\rangle = \langle\sin\tilde{\theta}\rangle = 0, \quad \langle\cos\tilde{\alpha}\rangle = e^{-\sigma_\alpha^2/2}, \quad \langle\cos\tilde{\theta}\rangle = e^{-\sigma_\theta^2/2} \qquad \text{A-1}$$

where σ_α^2 and σ_θ^2 are the variances in α and θ respectively.

If we write

$$x_m = x + \tilde{x} = (r + \tilde{r})\cos(\alpha + \tilde{\alpha})\cos(\theta + \tilde{\theta}), \qquad \text{A-2}$$

then expand A-2 using elementary trigonometric identities, and apply A-1, we find:

$$\langle\tilde{x}\rangle|_{r,\alpha,\beta} = r\cos\alpha\cos\theta\left(e^{-\sigma_\alpha^2/2-\sigma_\theta^2/2} - 1\right). \qquad \text{A-3}$$

The authors of [1], in presenting the derivation of the two-dimensional de-biasing formulae, apparently assume the two-dimensional analogue of

$$\langle\tilde{x}\rangle|_{r_m,\alpha_m,\beta_m} = \left\langle\langle\tilde{x}\rangle|_{r,\alpha,\beta}\right\rangle|_{r_m,\alpha_m,\beta_m} \qquad \text{A-4}$$

where the outer average on the right-hand side uses the same assumptions about the probability distributions of the polar errors as the inner average. If we substitute $r = r_m - \tilde{r}$ etc. into A-3 and use A-4, we get:

$$\langle \tilde{x} \rangle\Big|_{r_m,\alpha_m,\theta_m} = r_m \cos\alpha_m \cos\theta_m \left(e^{-\sigma_\alpha^2 - \sigma_\theta^2} - e^{-\sigma_\alpha^2/2 - \sigma_\theta^2} \right) \qquad \text{A-5}$$

Equation A-5 and its y and z counterparts lead directly to 2.4-3, while 2.4-4 is derived in an analogous way. (Note that the r, α, θ in equations 2.4-3 and 2.4-4 refer to the measured coordinates, which are denoted r_m, α_m, θ_m in this Annex.)

The trick represented by A-4 is not entirely convincing, because the calculation of the outer average assumes a probability distribution of the errors conditioned on the measured position that, if it were true, could be used directly to find $\langle \tilde{x} \rangle\Big|_{r_m,\alpha_m,\theta_m}$ (and would give us an answer that is clearly wrong). On the other hand, the numerical studies described in [1] indicate that this de-biasing scheme (in its two-dimensional version) works as advertised; that is, the transformed coordinates are apparently unbiased, and the covariance matrix is consistent with the spread of the errors. One hopes that there exists, somewhere, a more rigorous derivation of these equations.

B. The Munkres Algorithm

The following presentation of the Munkres algorithm, as modified by Burgeois and Lassalle, is based on the presentation in [2]. Blackman's presentation of it is incomplete in that it does not account for cases in which it is impossible to make as many associations as the lesser of the number of measurements and the number of tracks. (A program based directly on Blackman's presentation would in fact crash in such cases.) This deficiency is repaired here. Also, the steps are here modified to correspond more closely to traditional structured programming, as opposed to having numerous entangled GOTOs.

```
Let r = the number of rows, and c = the number of columns, of the assignment matrix;
Let a,b,d,m,n be integers;
Let X = a number that is much greater than the largest number in the assignment matrix;
Let W be a (numerical) matrix with r rows and c columns;
For m running from 1 to r, for n running from 1 to c,
        If the (m,n) element of the assignment matrix has a number,
                Set W(m,n) to this number;
        Otherwise,
                Set W(m,n) to X;
Let S and T be boolean matrices with r rows and c columns, with all elements initially "false";
Let R be an r-component boolean vector, initially all "false";
Let C be a c-component boolean vector, initially all "false";
If r is less than or equal to c,
        For m running from 1 to r,
                Let s = minimum(W(m,1), W(m,2), ..., W(m,c));
                For n running from 1 to c,
                        Decrease W(m,n) by s;
```

```
If c is less than or equal to r,
        For n running from 1 to c,
                Let s = minimum(W(1,n), W(2,n), ..., W(r,n));
                For m running from 1 to r,
                        Decrease W(m,n) by s;
For m running from 1 to r, for n running from 1 to c,
        If W(m,n)=0 AND NOT((S(m,1) OR S(m,2) OR ... OR S(m,c)) OR (S(1,n) OR ...
OR S(r,n))),
                Set S(m,n) to "true";
While the number of "true" elements of S is less than minimum(r,c),
        For n running from 1 to c,
                If S(1,n) OR ... OR S(r,n),
                        Set C(n) to "true";
        Repeat:
                If for all (m,n), W(m,n)>0 OR R(m) OR C(n),
                        Let h = the smallest value of W(m,n) for which NOT(R(m) OR C(n));
                        For m running from 1 to r,
                                If R(m),
                                        For n running from 1 to c,
                                                Increase W(m,n) by h;
                        For n running from 1 to c,
                                If NOT C(n),
                                        For m running from 1 to r,
                                                Decrease W(m,n) by h;
                Set a and b to any value such that W(a,b)=0  AND NOT(R(a) OR C(b));
                Set T(a,b) to "true";
                If S(a,1) OR S(a,2) OR ... OR S(a,c),
                        Let d = the number such that S(a,d);
                        Set R(a) to "true";
                        Set C(d) to "false";
        ...until NOT(S(a,1) OR S(a,2) OR ... OR S(a,c));
        While S(1,b) OR S(2,b) OR ... OR S(r,b),
                Let d = the number such that S(d,b);
                Set S(d,b) to "false";
                Set S(a,b) to "true";
                Let a = d;
                Let b = the number such that T(a,b);
        Set S(a,b) to "true";
        Set all elements of T, R, and C to "false";
For m running from 1 to r, for n running from 1 to c,
        If the (m,n) component of the original assignment matrix does not have a number,
                Set S(m,n) to "false";
Output S;
End.
```

The algorithm is now finished. The "true" elements of S mark the optimal association decisions.

Distribution List

DREA DOCUMENT DISTRIBUTION LIST

Document No.:

LIST PART 1: CONTROLLED BY DREA LIBRARY

2 DREA LIBRARY FILE COPIES
3 DREA LIBRARY (SPARES)

1 B. Chalmers
1 S. Flemming
1 J. S. Kennedy
1 B. McArthur
1 G. Mellema
1 D. Peters
1 W. Roger

12 TOTAL LIST PART 1

LIST PART 2: DISTRIBUTED BY DRDIM 3

1 NDHQ/ CRAD/ DRDIM 3
(scanned and stored as black & white image, low resolution
- laser reprints available on request)

1 TOTAL LIST PART 2

12 TOTAL COPIES REQUIRED

Original document held by DREA Drafting Office

Any requests by DREA staff for extra copies of this document should be directed to the DREA LIBRARY.

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research Establishment Atlantic

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable).

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title).

A Practical Guide to Level One Data Fusion Algorithms

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

Peters, Daniel J.

5. DATE OF PUBLICATION (month and year of publication of document)

December 2001

6a. NO OF PAGES (total containing information Include Annexes, Appendices, etc).

56

6b. NO OF REFS (total cited in document)

11

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered).

TECHNICAL MEMORANDUM

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include address).

Defence Research Establishment Atlantic
PO Box 1012
Dartmouth, NS, Canada B2Y 3Z7

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant).

11ba

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written).

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DREA TM 2001-201

10b. OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11) is possible, a wider announcement audience may be selected).

unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual)

Level one data fusion is the process of combining data in order to track and classify individual entities. This document introduces the basic concepts and presents a core selection of standard algorithms, such as the Kalman Filter, the Interacting Multiple Model (IMM) filter, the Probabilistic Data Association Filter (PDAF) and its Joint variant, the Munkres algorithm for Nearest Neighbour (NN) association, and Multiple-Hypothesis Tracking (MHT), among others. It is intended to serve as a convenient one-stop repository of algorithms.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title).

data fusion, data association, target tracking